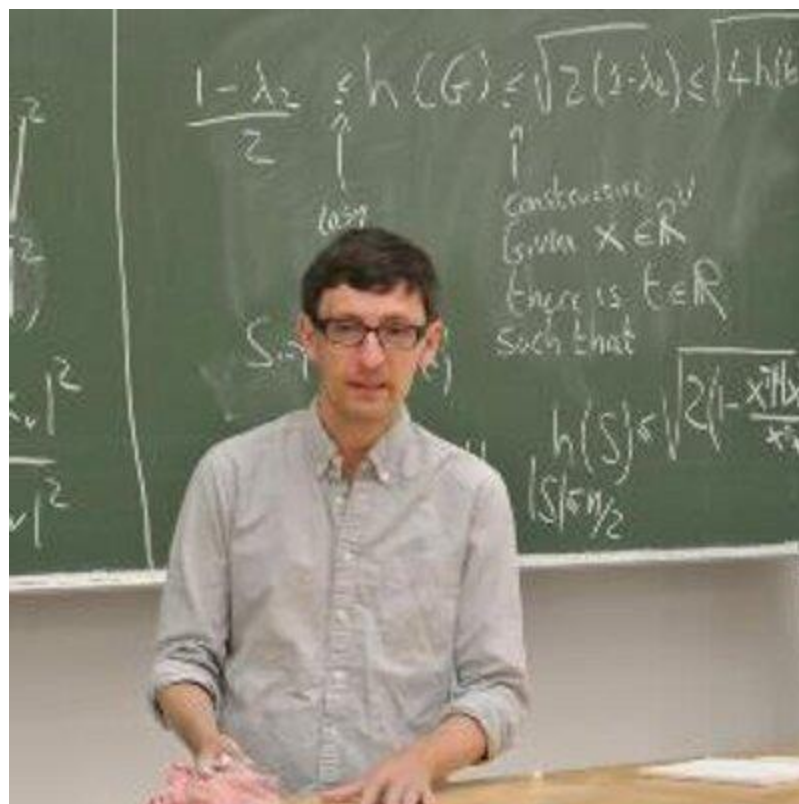


Three Key Ideas in the LLM Revolution

Janardhan Kulkarni,
The Algorithms Group,
MSR, Redmond.

Remembering Luca



How I got into LLMs

- Background in TCS: Approximation and Online Algorithms
- Inspired by GPT3/Protein folding papers around 2019/20
- Lead Differentially Private Training of LLMs at MS/MSR
 - Our DP trained LMs power many critical applications such as Outlook, Word, Onenote, etc, and are largest deployed transformer models in terms of volume in MS (as of 2023😊).
- Current Projects in the Algorithms Group: Post-transformers/ long sequence modelling, improving reasoning/planning

Goals of the workshop

- **Start a conversation in our community.** Invite beginning graduate students to explore the workings of LLMs/DNNs, and the evolution of intelligence/cognition
 - spend 20-30% of time thinking/reading
- **Timely:** Many things are standardized. Model architecture, training recipes, optimization algorithms, etc., have withstood test of time. Perfect time to investigate theoretically/algorithmically
- Industry has taken care of problems that can be solved by data and scale: rest of us can focus on truly new ideas. **There are plenty.**

Quantization of GPT3/4 via Discrepancy

- **Quantization problem: Given 32-bit floats corresponding to a GPT model, “round” it to 4-bit values keeping the performance.**
- At MSR, we designed an algorithm inspired by online/streaming discrepancy minimization problem
- More about Quantization: See GPTQ paper (<https://arxiv.org/abs/2210.17323>)
- More about online discrepancy minimization: see our talk on Friday (<https://arxiv.org/abs/2308.01406>)
- *Open Problem: Design a streaming/online algorithm for finding a basic feasible solution x^* of an LP that has smallest ℓ_2 distance to a given feasible solution y (that is good on GPUs)*

Other talks on LLMs at STOC

- Keynote by Jakub
- Paper on hallucinations by Kalai and Vempala

Brief history of LLM revolution

introduce key ideas/concepts needed to follow remaining topics

Agenda

- **Going from supervised learning to unsupervised or self-supervised learning and in-context learning**

GPT2 and 3 papers, CLIP/unCLIP papers (Text and Image co-representation), Diffusion Models

- **Standardization of the training and model architectures**

Attention is all you need paper

- **Scaling**

Scaling laws for neural language models, Chinchilla papers

Disclaimer

I will not be exhaustive in my citations of the works.
Please see the cited papers to for more comprehensive/chronological evolution of ideas.

Biased view: How I, as a TCS person, sees the LLM revolution.

All Build on Deep Learning Revolution

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet ILSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

1 Introduction

Current approaches to object recognition make essential use of machine learning methods. To improve their performance, we can collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting. Until recently, datasets of labeled images were relatively small — on the order of tens of thousands of images (e.g., NORB [16], Caltech-101/256 [8, 9], and CIFAR-10/100 [12]). Simple recognition tasks can be solved quite well with datasets of this size, especially if they are augmented with label-preserving transformations. For example, the current-best error rate on the MNIST digit-recognition task ($<0.3\%$) approaches human performance [4]. But objects in realistic settings exhibit considerable variability, so to learn to recognize them it is necessary to use much larger training sets. And indeed, the shortcomings of small image datasets have been widely recognized (e.g., Pinto et al. [21]), but it has only recently become possible to collect labeled datasets with millions of images. The new larger datasets include LabelMe [23], which consists of hundreds of thousands of fully-segmented images, and ImageNet [6], which consists of over 15 million labeled high-resolution images in over 22,000 categories.

To learn about thousands of objects from millions of images, we need a model with a large learning capacity. However, the immense complexity of the object recognition task means that this problem cannot be specified even by a dataset as large as ImageNet, so our model should also have lots of prior knowledge to compensate for all the data we don't have. Convolutional neural networks (CNNs) constitute one such class of models [16, 11, 13, 18, 15, 22, 26]. Their capacity can be controlled by varying their depth and breadth, and they also make strong and mostly correct assumptions about the nature of images (namely, stationarity of statistics and locality of pixel dependencies). Thus, compared to standard feedforward neural networks with similarly-sized layers, CNNs have much fewer connections and parameters and so they are easier to train, while their theoretically-best performance is likely to be only slightly worse.

- Deepnets are better at learning **feature representation**
- No need for **hand-tuned features** (SVMs)
- DNNs need lots of data and compute

Basics of Deep Learning

- Handwritten recognition: hello world of DL
- Dataset: MNIST
- Goal: Recognise which number is in the picture?

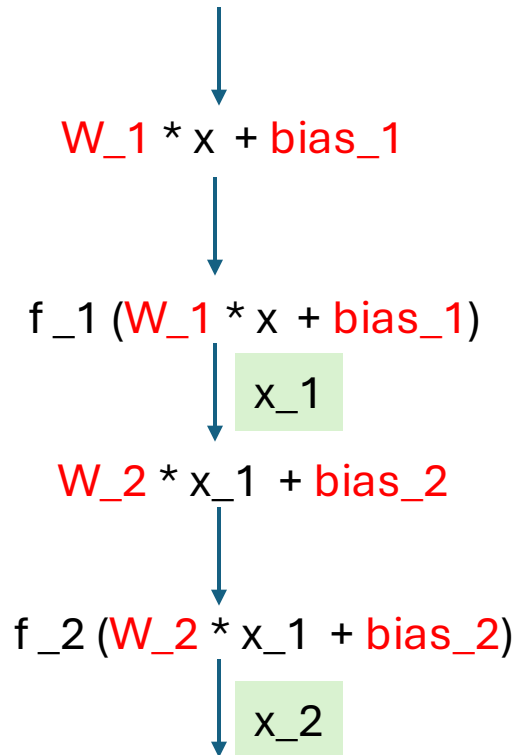


Input: 28 * 28 matrix or a vector representing a number; each value represents greyscale value of the pixel.

Output: predict which of the following 10 digits it represents

Deep Learning Way

Input: 784-dimensional vector x .



x_2 is a probability distribution over 10 digits

W_1 and W_2 are matrices of appropriate dimensions

Bias vectors are also of right dimensions

f_1 and f_2 are non-linear functions applied element wise

* ReLU (a): $\max(0, a)$

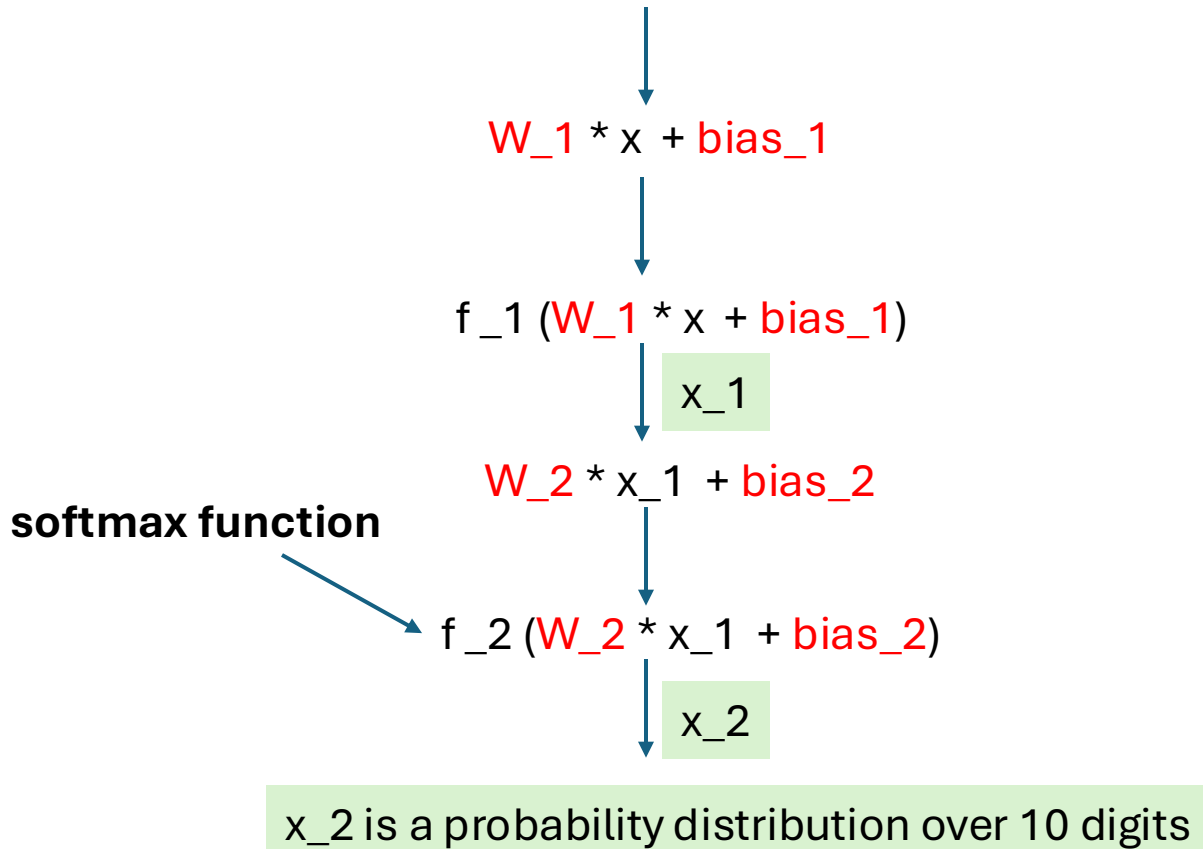
* softmax function:

Formally, the standard (unit) softmax function $\sigma: \mathbb{R}^K \rightarrow (0, 1)^K$, where $K \geq 1$, takes a vector $\mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K$ and computes each component of vector $\sigma(\mathbf{z}) \in (0, 1)^K$ with

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}.$$

Deep Learning Way

Input: 784-dimensional vector x .



W_1 and W_2 are matrices of appropriate dimensions

Bias vectors are also of right dimensions

f_1 and f_2 are non-linear functions applied element wise

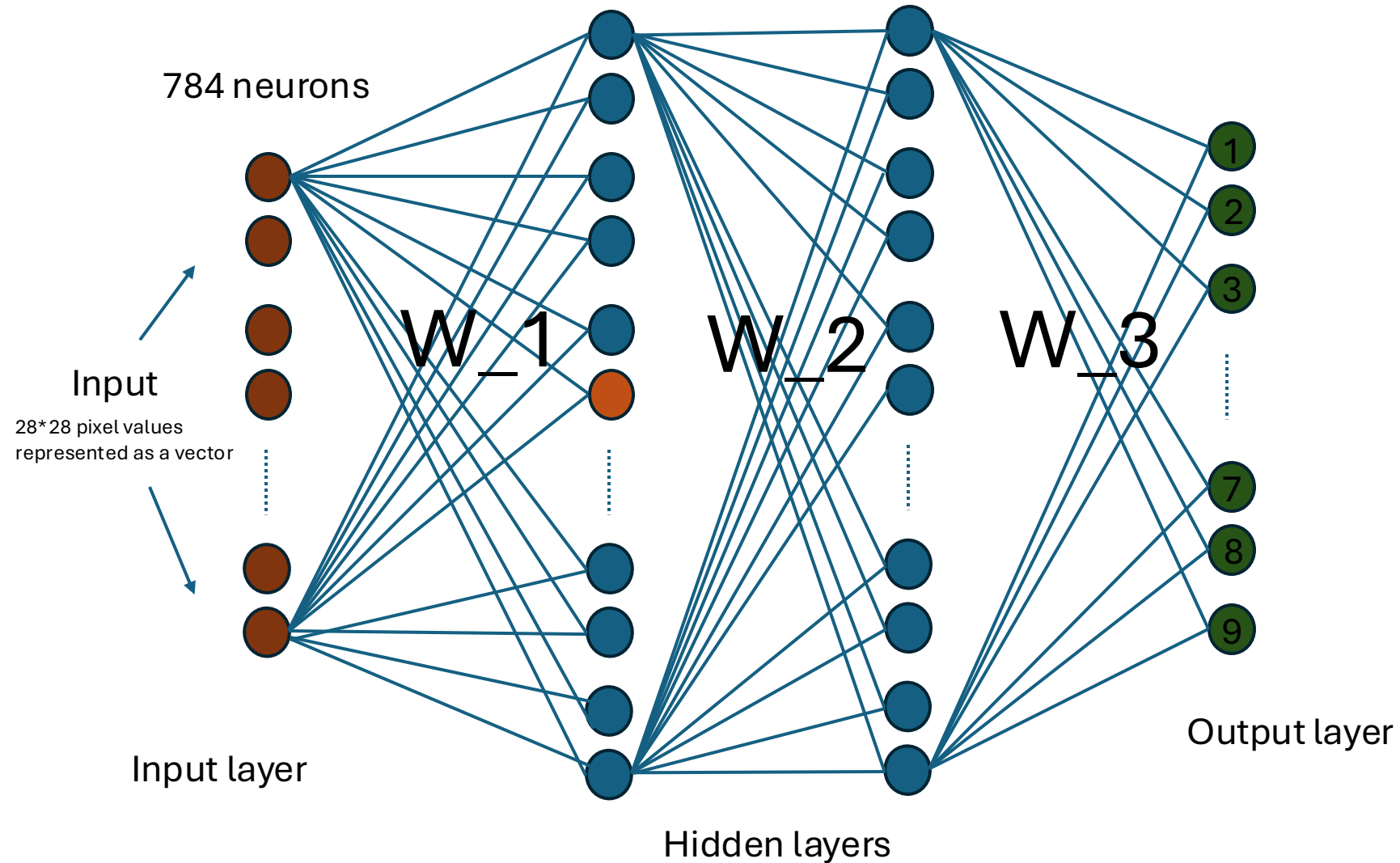
* ReLU (a): $\max(0, a)$

* softmax function:

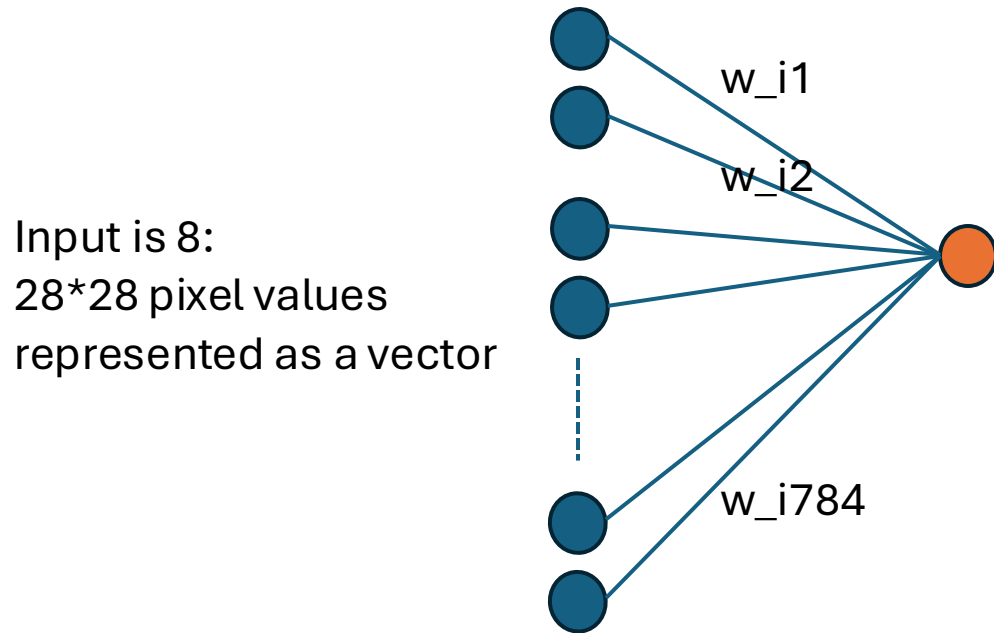
Formally, the standard (unit) softmax function $\sigma: \mathbb{R}^K \rightarrow (0, 1)^K$, where $K \geq 1$, takes a vector $\mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K$ and computes each component of vector $\sigma(\mathbf{z}) \in (0, 1)^K$ with

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}.$$

Fully connected feed forward networks (some times called MLPs)



What is each neuron doing?



Neurons are tiny computational units



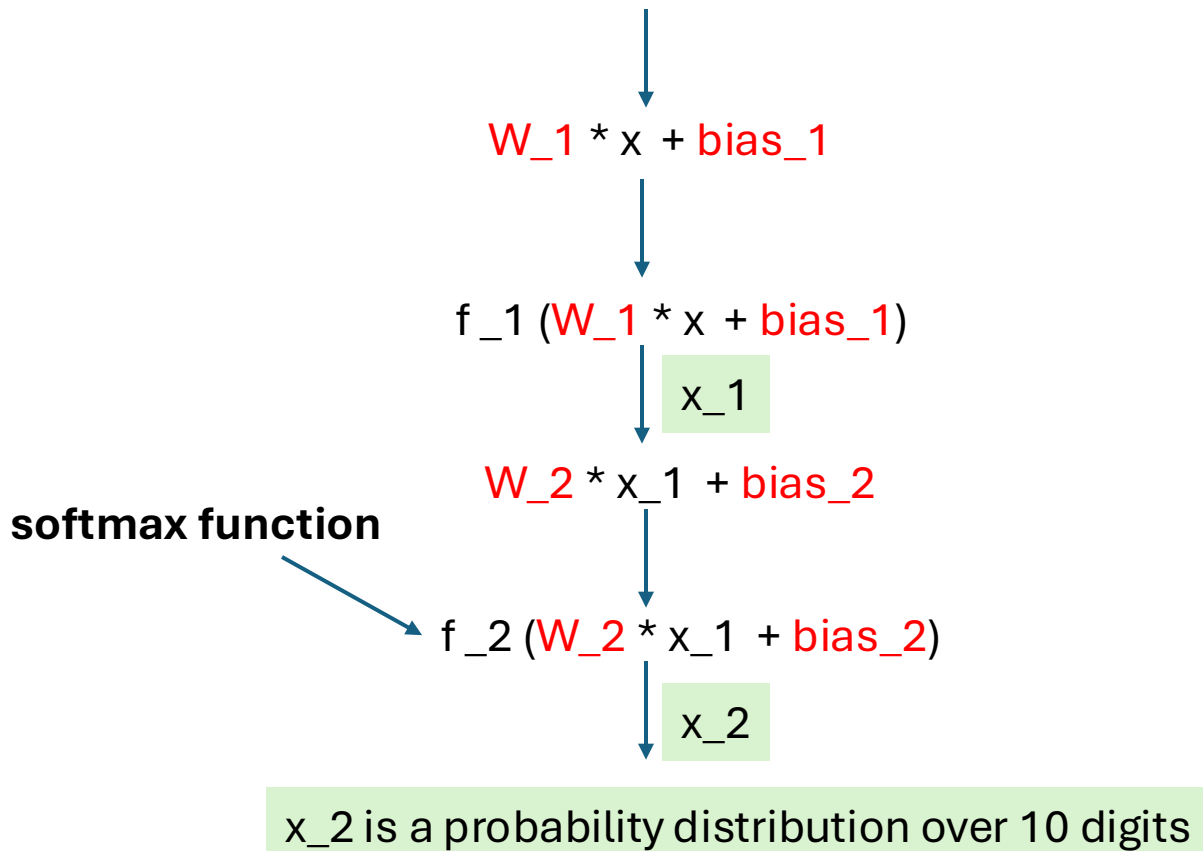
$$\text{ReLU}(w_{i1} * x_1 + w_{i2} * x_2 \dots + w_{in} * x_n - \text{bias}_i)$$



Activations: the output after applying activation functions

Deep Learning = Sequence of matmuls!

Input: 784-dimensional vector x .



W_1 and W_2 are matrices of appropriate dimensions

Bias vectors are also of right dimensions

f_1 and f_2 are non-linear functions applied element wise

* ReLU (a): $\max(0, a)$

* softmax function:

Formally, the standard (unit) softmax function $\sigma: \mathbb{R}^K \rightarrow (0, 1)^K$, where $K \geq 1$, takes a vector $\mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K$ and computes each component of vector $\sigma(\mathbf{z}) \in (0, 1)^K$ with

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}.$$

What is learning?

Input: 784-dimensional vector x .

$$W_1 * x + \text{bias}_1$$

$$f_1(W_1 * x + \text{bias}_1)$$

x_1

$$W_2 * x_1 + \text{bias}_2$$

softmax function

$$f_2(W_2 * x_1 + \text{bias}_2)$$

x_2

x_2 is a probability distribution over 10 digits

Find good weight matrices W_1 , W_2 and bias vectors

Learn from (labelled & lots) data

1. Find a large set of labelled data

The MNIST database contains 60,000 training images and 10,000 testing images

2. Associate a loss function

If the output predicted by the algorithm is correct, the loss should be small. Otherwise, it should be high.

- Squared loss function
- Cross entropy loss function →

The cross-entropy loss for a single prediction can be defined as:

$$L = - \sum_{i=1}^N y_i \log(p_i)$$

where:

- N is the number of classes (in the case of language models, this is the size of the vocabulary).
- y_i is the actual binary indicator (0 or 1) if class i is the correct class.
- p_i is the predicted probability for class i .

→ $-\log(\text{prob}(\text{true label}))$

3. Optimize the loss function via gradient descent type algorithms: SGD, Adam.

Notes on Loss Function

- Our data is independent samples from an unknown distribution
- We treat neural network as a representation of this unknown distribution
- But neural network is a function of its parameters
- So, find the parameters that bring our neural network distribution closer to the true unknown distribution given through samples

Maximizing the likelihood function

=

Maximizing the log likelihood function

=

Minimizing the – negative log likelihood function

Cross-entropy

[Article](#) [Talk](#)

From Wikipedia, the free encyclopedia

In [information theory](#), the **cross-entropy** between two [probability distributions](#) p and q , over the same underlying set of events, measures the average number of [bits](#) needed to identify an event drawn from the set when the coding scheme used for the set is optimized for an estimated probability distribution q , rather than the true distribution p .

Definition [\[edit \]](#)

The cross-entropy of the distribution q relative to a distribution p over a given set is defined as follows:

$$H(p, q) = - \mathbb{E}_p[\log q],$$

where $E_p[\cdot]$ is the [expected value](#) operator with respect to the distribution p .

Notes on Loss Function

- Our data is independent samples from an unknown distribution

Relation to **maximum** likelihood [\[edit \]](#)

The cross entropy arises in classification problems when introducing a logarithm in the guise of the [log-likelihood](#) function.

The section is concerned with the subject of estimation of the probability of different possible discrete outcomes. To this end, denote a parametrized family of distributions by q_θ , with θ subject to the optimization effort. Consider a given finite sequence of N values x_i from a training set, obtained from [conditionally independent](#) sampling. The likelihood assigned to any considered parameter θ of the model is then given by the product over all probabilities $q_\theta(X = x_i)$. Repeated occurrences are possible, leading to equal factors in the product. If the count of occurrences of the value equal to x_i (for some index i) is denoted by $\#x_i$, then the frequency of that value equals $\#x_i / N$. Denote the latter by $p(X = x_i)$, as it may be understood as empirical approximation to the probability distribution underlying the scenario. Further denote by $PP := e^{H(p, q_\theta)}$ the [perplexity](#), which can be seen to equal $\prod_{x_i} q_\theta(X = x_i)^{-p(X=x_i)}$ by the [calculation rules for the logarithm](#), and where the product is over the values without double counting. So

$$\mathcal{L}(\theta; \mathbf{x}) = \prod_i q_\theta(X = x_i) = \prod_{x_i} q_\theta(X = x_i)^{\#x_i} = PP^{-N} = e^{-N \cdot H(p, q_\theta)}$$

or

$$\log \mathcal{L}(\theta; \mathbf{x}) = -N \cdot H(p, q_\theta).$$

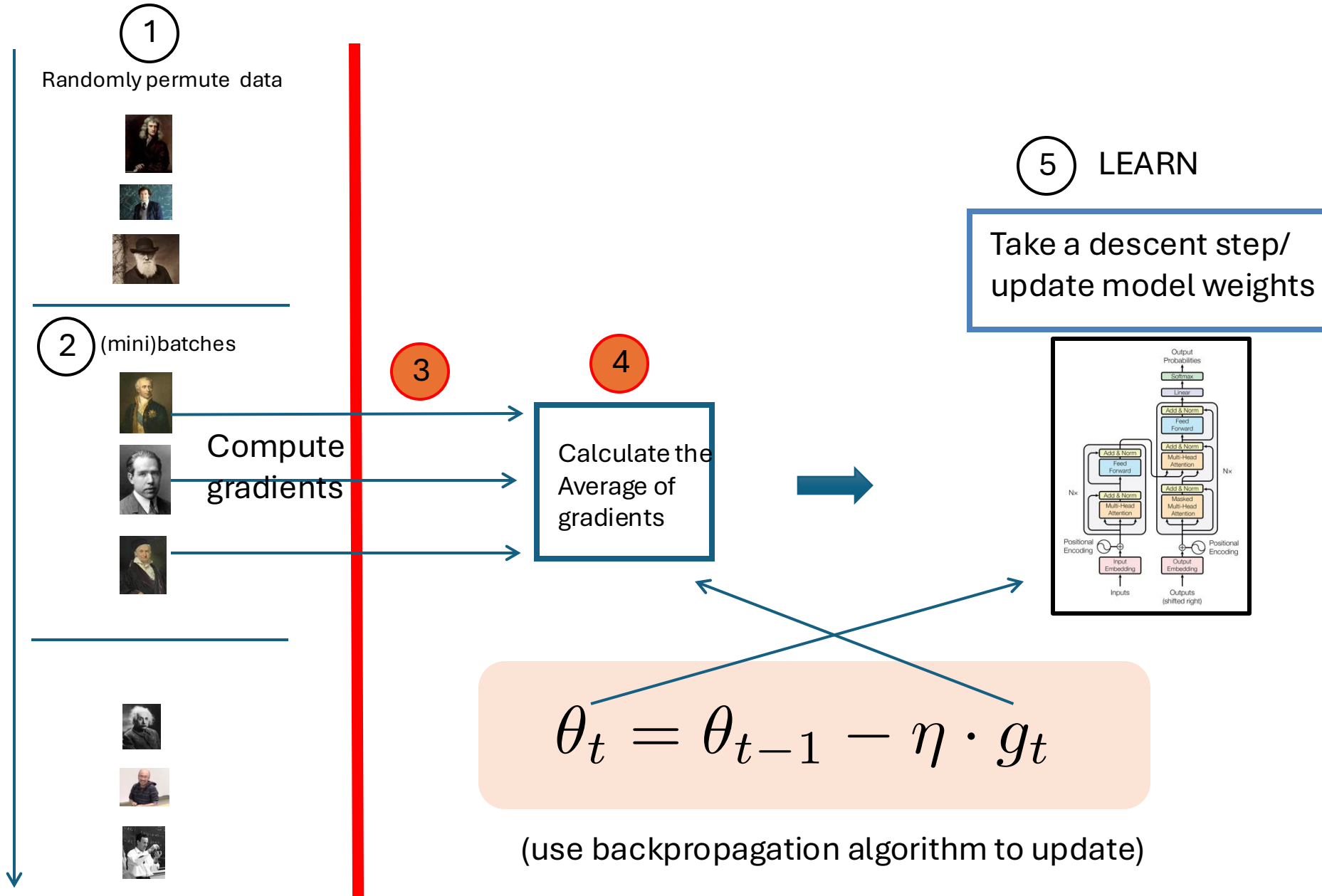
Minimizing the – negative log likelihood function

The cross-entropy of the distribution q relative to a distribution p over a given set is defined as follows:

$$H(p, q) = -\mathbb{E}_p[\log q],$$

where $E_p[\cdot]$ is the [expected value](#) operator with respect to the distribution p .

Training Deep Models via SGD/Adam



Neural Networks Really work! “AlexNet Moment”

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called “dropout” that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

1 Introduction

Current approaches to object recognition make essential use of machine learning methods. To improve their performance, we can collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting. Until recently, datasets of labeled images were relatively small — on the order of tens of thousands of images (e.g., NORB [16], Caltech-101/256 [8, 9], and CIFAR-10/100 [12]). Simple recognition tasks can be solved quite well with datasets of this size, especially if they are augmented with label-preserving transformations. For example, the current-best error rate on the MNIST digit-recognition task (<0.3%) approaches human performance [4]. But objects in realistic settings exhibit considerable variability, so to learn to recognize them it is necessary to use much larger training sets. And indeed, the shortcomings of small image datasets have been widely recognized (e.g., Pinto et al. [21]), but it has only recently become possible to collect labeled datasets with millions of images. The new larger datasets include LabelMe [23], which consists of hundreds of thousands of fully-segmented images, and ImageNet [6], which consists of over 15 million labeled high-resolution images in over 22,000 categories.

To learn about thousands of objects from millions of images, we need a model with a large learning capacity. However, the immense complexity of the object recognition task means that this problem cannot be specified even by a dataset as large as ImageNet, so our model should also have lots of prior knowledge to compensate for all the data we don’t have. Convolutional neural networks (CNNs) constitute one such class of models [16, 11, 13, 18, 15, 22, 26]. Their capacity can be controlled by varying their depth and breadth, and they also make strong and mostly correct assumptions about the nature of images (namely, stationarity of statistics and locality of pixel dependencies). Thus, compared to standard feedforward neural networks with similarly-sized layers, CNNs have much fewer connections and parameters and so they are easier to train, while their theoretically-best performance is likely to be only slightly worse.



Contents

- (Top)
- Historic context
- Network design
- Influence
- References

Search Wikipedia

AlexNet

Article Talk

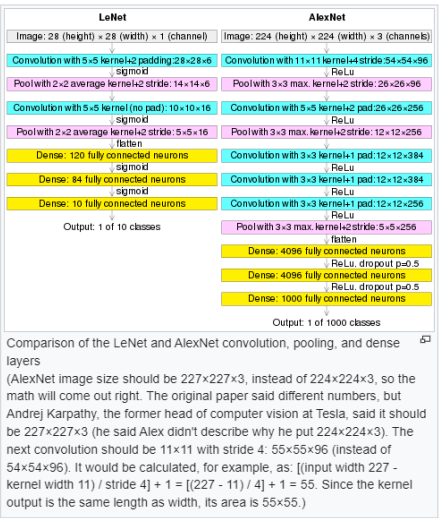
From Wikipedia, the free encyclopedia

AlexNet is the name of a **convolutional neural network** (CNN) architecture, designed by **Alex Krizhevsky** in collaboration with **Ilya Sutskever** and **Geoffrey Hinton**, who was Krizhevsky’s Ph.D. advisor at the University of Toronto.^{[1][2]}

AlexNet competed in the **ImageNet Large Scale Visual Recognition Challenge** on September 30, 2012.^[3] The network achieved a top-5 error of 15.3%, more than 10.8 percentage points lower than that of the runner up. The original paper’s primary result was that the depth of the model was essential for its high performance, which was computationally expensive, but made feasible due to the utilization of **graphics processing units** (GPUs) during training.^[2]

Historic context

AlexNet was not the first fast GPU-implementation of a CNN to win an image recognition contest. A CNN on GPU by K. Chellapilla et al. (2006) was 4 times faster than an equivalent implementation on CPU.^[4] A deep CNN of **Dan Cireşan** et al. (2011) at **IDSIA** was already 60 times faster^[5] and outperformed predecessors in August 2011.^[6] Between May 15, 2011, and September 10, 2012, their CNN won no fewer than four image competitions.^{[7][8]} They also significantly improved on the best performance in the literature for multiple image **databases**.^[9]



Create account Log in

Appearance

- Text
- ☐ Small
- ☒ Standard
- ☐ Large
- Width
- ☒ Standard
- ☐ Wide

14 million samples
annotated by hand

ImageNet

🌐 13 languages ▾

Article [Talk](#)

[Read](#) [Edit](#) [View history](#) [Tools](#) ▾

From Wikipedia, the free encyclopedia

(Redirected from [ImageNet Large Scale Visual Recognition Challenge](#))

The **ImageNet** project is a large visual [database](#) designed for use in [visual object recognition software](#) research. More than 14 million^{[1][2]} images have been hand-annotated by the project to indicate what objects are pictured and in at least one million of the images, bounding boxes are also provided.^[3] ImageNet contains more than 20,000 categories,^[2] with a typical category, such as "balloon" or "strawberry", consisting of several hundred images.^[4] The database of annotations of third-party image URLs is freely available directly from ImageNet, though the actual images are not owned by ImageNet.^[5] Since 2010, the ImageNet project runs an annual software contest, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), where software programs compete to correctly classify and detect objects and scenes. The challenge uses a "trimmed" list of one thousand non-overlapping classes.^[6]

Significance for deep learning [\[edit \]](#)

On 30 September 2012, a [convolutional neural network](#) (CNN) called [AlexNet](#)^[7] achieved a top-5 error of 15.3% in the ImageNet 2012 Challenge, more than 10.8 percentage points lower than that of the runner up. This was made feasible due to the use of [graphics processing units](#) (GPUs) during training,^[7] an essential ingredient of the [deep learning](#) revolution. According to *The Economist*, "Suddenly people started to pay attention, not just within the AI community but across the technology industry as a whole."^{[4][8][9]}

In 2015, AlexNet was outperformed by Microsoft's [very deep CNN](#) with over 100 layers, which won the ImageNet 2015 contest.^[10]

History of the database [\[edit \]](#)

AI researcher [Fei-Fei Li](#) began working on the idea for ImageNet in 2006. At a time when most AI research focused on models and algorithms, Li wanted to expand and improve the data available to train AI algorithms.^[11] In 2007, Li met with Princeton professor [Christiane Fellbaum](#), one of the creators of [WordNet](#), to discuss the project. As a result of this meeting, Li went on to build ImageNet starting from the word database of WordNet and using many of its features.^[12]

As an assistant professor at Princeton, Li assembled a team of researchers to work on the ImageNet project. They used [Amazon Mechanical Turk](#) to help with the classification of images.^[12]

They presented their database for the first time as a poster at the 2009 [Conference on Computer Vision and Pattern Recognition](#) (CVPR) in Florida.^{[12][13][14]}

Dataset [\[edit \]](#)

ImageNet [crowdsources](#) its annotation process. Image-level annotations indicate the presence or absence of an object class in an image, such as "there are tigers in this image" or "there are no tigers in this image". Object-level annotations provide a bounding box around the (visible part of the) indicated object. ImageNet uses a variant of the broad [WordNet](#) schema to categorize objects, augmented with 120 categories of [dog breeds](#) to showcase fine-grained classification.^[6] One downside of WordNet use is the categories may be more "elevated" than would be optimal for ImageNet: "Most people are more interested in Lady Gaga or the iPod Mini than in this rare kind of [diplodocus](#)."^[clarification needed] In 2012 ImageNet was the world's largest academic user of [Mechanical Turk](#). The average worker identified 50 images per minute.^[2]

Subsets of the dataset [\[edit \]](#)

There are various subsets of the ImageNet dataset used in various context. One of the most highly used subset of ImageNet is the "ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012-2017 image classification and localization dataset". This is also referred to in the research literature as ImageNet-1K or ILSVRC2017, reflecting the original ILSVRC challenge that involved 1,000 classes. ImageNet-1K contains 1,281,167 training images, 50,000 validation images and 100,000 test images.^[15] The full original dataset is referred to as ImageNet-21K. ImageNet-21k contains 14,197,122 images divided into 21,841 classes. Some papers round this up and name it ImageNet-22k.^[16]

Prior to LLM revolution (< 2014)

- **Supervised learning**
 - create labelled dataset
 - Imagenet creator Fei Fei Li is celebrated for this
- Train a **task-specific** DNN
 - identify images, translation, sentiment analysis, so on.
- Lots of design parameters: Model architecture, data augmentation tricks, learning algorithms etc



Already spectacular success

Bert, GPT2/GPT3 Papers Changed Something Fundamental

Language Models are Unsupervised Multitask Learners

Alec Radford^{*1} Jeffrey Wu^{*1} Rewon Child¹ David Luan¹ Dario Amodei^{**1} Ilya Sutskever^{**1}

Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the answers generated by the language model reach 55 F1 on the CoQA dataset - matching or exceeding the performance of 3 out of 4 baseline systems without using the 127,000+ training examples. The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2, is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText. Samples from the model reflect these improvements and contain coherent paragraphs of text. These findings suggest a promising path towards building language processing systems which learn to perform tasks from their naturally occurring demonstrations.

1. Introduction

Machine learning systems now excel (in expectation) at tasks they are trained for by using a combination of large datasets, high-capacity models, and supervised learning (Krizhevsky et al., 2012) (Sutskever et al., 2014) (Amodei et al., 2016). Yet these systems are brittle and sensitive to slight changes in the data distribution (Recht et al., 2018) and task specification (Kirkpatrick et al., 2017). Current systems are better characterized as narrow experts rather than

competent generalists. We would like to move towards more general systems which can perform many tasks – eventually without the need to manually create and label a training dataset for each one.

The dominant approach to creating ML systems is to collect a dataset of training examples demonstrating correct behavior for a desired task, train a system to imitate these behaviors, and then test its performance on independent and identically distributed (IID) held-out examples. This has served well to make progress on narrow experts. But the often erratic behavior of captioning models (Lake et al., 2017), reading comprehension systems (Jia & Liang, 2017), and image classifiers (Alcorn et al., 2018) on the diversity and variety of possible inputs highlights some of the shortcomings of this approach.

Our suspicion is that the prevalence of single task training on single domain datasets is a major contributor to the lack of generalization observed in current systems. Progress towards robust systems with current architectures is likely to require training and measuring performance on a wide range of domains and tasks. Recently, several benchmarks have been proposed such as GLUE (Wang et al., 2018) and decaNLP (McCann et al., 2018) to begin studying this.

Multitask learning (Caruana, 1997) is a promising framework for improving general performance. However, multitask training in NLP is still nascent. Recent work reports modest performance improvements (Yogatama et al., 2019) and the two most ambitious efforts to date have trained on a total of 10 and 17 (dataset, objective) pairs respectively (McCann et al., 2018) (Bowman et al., 2018). From a meta-learning perspective, each (dataset, objective) pair is a single training example sampled from the distribution of datasets and objectives. Current ML systems need hundreds to thousands of examples to induce functions which generalize well. This suggests that multitask training may need just as many effective training pairs to realize its promise with current approaches. It will be very difficult to continue to scale the creation of datasets and the design of objectives to the degree that may be required to brute force our way there with current techniques. This motivates exploring additional setups for performing multitask learning.

The current best performing systems on language tasks

Language Models are Few-Shot Learners

Tom B. Brown*	Benjamin Mann*	Nick Ryder*	Melanie Subbiah*	
Jared Kaplan†	Prafulla Dhariwal	Arvind Neelakantan	Pranav Shyam	Girish Sastry
Amanda Askell	Sandhini Agarwal	Ariel Herbert-Voss	Gretchen Krueger	Tom Henighan
Rewon Child	Aditya Ramesh	Daniel M. Ziegler	Jeffrey Wu	Clemens Winter
Christopher Hesse	Mark Chen	Eric Sigler	Mateusz Litwin	Scott Gray
Benjamin Chess	Jack Clark	Christopher Berner		
Sam McCandlish	Alec Radford	Ilya Sutskever	Dario Amodei	

OpenAI

Abstract

Recent work has demonstrated substantial gains on many NLP tasks and benchmarks by pre-training on a large corpus of text followed by fine-tuning on a specific task. While typically task-agnostic in architecture, this method still requires task-specific fine-tuning datasets of thousands or tens of thousands of examples. By contrast, humans can generally perform a new language task from only a few examples or from simple instructions – something which current NLP systems still largely struggle to do. Here we show that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even reaching competitiveness with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x more than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves strong performance on many NLP datasets, including translation, question-answering, and cloze tasks, as well as several tasks that require on-the-fly reasoning or domain adaptation, such as unscrambling words, using a novel word in a sentence, or performing 3-digit arithmetic. At the same time, we also identify some datasets where GPT-3's few-shot learning still struggles, as well as some datasets where GPT-3 faces methodological issues related to training on large web corpora. Finally, we find that GPT-3 can generate samples of news articles which human evaluators have difficulty distinguishing from articles written by humans. We discuss broader societal impacts of this finding and of GPT-3 in general.

*Equal contribution

†Johns Hopkins University, OpenAI

Author contributions listed at end of paper.

**Equal contribution ¹OpenAI, San Francisco, California, United States. Correspondence to: Alec Radford <alec@openai.com>.

arXiv:2005.14165v4 [cs.CL] 22 Jul 2020

GPT2 and 3 papers changed that paradigm (forever)...

Language Models are Unsupervised Multitask Learners

Alec Radford¹ Jeffrey Wu¹ Rewon Child¹ David Luan¹ Dario Amodei^{1*} Ilya Sutskever^{1,2}

Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the answers generated by the language model reach 55 F1 on the CoQA dataset – matching or exceeding the performance of 3 out of 4 baseline systems without using the 127,000+ training examples. The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2, is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText. Samples from the model reflect these improvements and contain coherent paragraphs of text. These findings suggest a promising path towards building language processing systems which learn to perform tasks from their naturally occurring demonstrations.

1. Introduction

Machine learning systems now excel (in expectation) at tasks they are trained for by using a combination of large datasets, high-capacity models, and supervised learning (Krizhevsky et al., 2012; Sutskever et al., 2014) (Anadek et al., 2016). Yet these systems are brittle and sensitive to slight changes in the data distribution (Recht et al., 2018) and task specification (Kirkpatrick et al., 2017). Current systems are better characterized as narrow experts rather than competent generalists. We would like to move towards more general systems which can perform many tasks – eventually without the need to manually create and label a training dataset for each one.

The dominant approach to creating ML systems is to collect a dataset of training examples demonstrating correct behavior for a desired task, train a system to imitate these behaviors, and then test its performance on independent and identically distributed (IID) held-out examples. This has served well to make progress on narrow experts. But the often erratic behavior of captioning models (Lake et al., 2017), reading comprehension systems (Ia & Liang, 2017), and image classifiers (Alcorn et al., 2018) on the diversity and variety of possible inputs highlights some of the shortcomings of this approach.

Our suspicion is that the prevalence of single task training on single domain datasets is a major contributor to the lack of generalization observed in current systems. Progress towards robust systems with current architectures is likely to require training and measuring performance on a wide range of domains and tasks. Recently, several benchmarks have been proposed such as GLUE (Wang et al., 2018) and decaNLP (McCam et al., 2018) to begin studying this.

Multitask learning (Caruana, 1997) is a promising framework for improving general performance. However, multitask training in NLP is still nascent. Recent work reports modest performance improvements (Yogatama et al., 2019) and the two most ambitious efforts to date have trained on a total of 10 and 17 (dataset, subject) pairs respectively (McCam et al., 2018) (Bosman et al., 2018). From a meta-learning perspective, each (dataset, subject) pair is a single training example sampled from the distribution of datasets and objectives. Current ML systems need hundreds to thousands of examples to induce functions which generalize well. This suggests that multitask training may need just as many effective training pairs to realize its promise with current approaches. It will be very difficult to continue to scale the creation of datasets and the design of objectives to the degree that may be required to brute force our way there with current techniques. This motivates exploring additional setups for performing multitask learning.

The current best performing systems on language tasks

Language Models are Few-Shot Learners				
Tom R. Brown [*]	Benjamin Mann [*]	Nick Ryder [*]	Melanie Subbiah [*]	
Jared Kaplan ¹	Prafulla Dhariwal	Arvind Neelakantan	Pranav Shyam	Girish Sastry
Amanda Askell	Sandhini Agarwal	Ariel Herbert-Voss	Gretchen Krueger	Tom Henighan
Rewon Child	Aditya Ramesh	Daniel M. Ziegler	Jeffrey Wu	Clemens Winter
Christopher Hesse	Mark Chen	Eric Sigler	Mateusz Litwin	Scott Gray
Benjamin Chess		Jack Clark	Christopher Berner	
Sam McCandlish	Alec Radford	Ilya Sutskever	Dario Amodei	
OpenAI				
Abstract				
<p>Recent work has demonstrated substantial gains on many NLP tasks and benchmarks by pre-training on a large corpus of text followed by fine-tuning on a specific task. While typically task-agnostic in architecture, this method still requires task-specific fine-tuning datasets of thousands or tens of thousands of examples. By contrast, humans can generally perform a new language task from only a few examples or from simple instructions – something which current NLP systems still largely struggle to do. Here we show that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even reaching competitiveness with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x more than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves strong performance on many NLP datasets, including translation, question-answering, and clone tasks, as well as several tasks that require on-the-fly reasoning or domain adaptation, such as unscrambling words, using a novel word in a sentence, or performing 3-digit arithmetic. At the same time, we also identify some datasets where GPT-3's few-shot learning still struggles, as well as some datasets where GPT-3 faces methodological issues related to training on large web corpora. Finally, we find that GPT-3 can generate samples of news articles which human evaluators have difficulty distinguishing from articles written by humans. We discuss broader societal impacts of this finding and of GPT-3 in general.</p>				

^{*}Equal contribution
¹Johns Hopkins University, OpenAI
Author contributions listed at end of paper.

- No more task specific training.
- We train a single model that will be used to solve any language task.
- Accomplished via “in-context learning”
- Give some examples of tasks you want to solve, and then ask your question to the model.


arXiv:2005.14165v4 [cs.CL] 22 Jul 2020

Incontext learning, fewshot/zeroshot learning

Example 5: Code Completion

Prompt:

python

 Copy code

```
# Function to add two numbers
def add(a, b):
    return a + b


# Function to subtract two numbers
def subtract(a, b):
    return a - b

# Function to multiply two numbers
def multiply(a, b):
    return a * b

# Function to divide two numbers
def divide(a, b):
```

Model Response:

python

 Copy code


```
    return a / b
```

Incontext learning, fewshot/zeroshot learning

Example 6: Grammar Correction

Prompt:

vbnet

 Copy code

Correct the grammar of the following sentences:

1. She don't like apples.

Correction: She doesn't like apples.

2. He go to school everyday.

Correction: He goes to school every day.


3. I can plays the guitar.

Correction: I can play the guitar.

4. They was very happy.

Model Response:

makefile

 Copy code


Correction: They were very happy.

Incontext learning, fewshot/zeroshot learning

Example 2: Translation

Prompt:

vbnet

 Copy code

Translate the following sentences from English to French:

1. The cat is on the roof.

Le chat est sur le toit.

2. I love to eat chocolate.

J'aime manger du chocolat.


3. She is reading a book.

Elle lit un livre.

4. The weather is nice today.

Model Response:

rust

 Copy code

Il fait beau aujourd'hui.



GPT2 and 3 papers changed that paradigm (forever)...

Language Models are Unsupervised Multitask Learners

Alec Radford¹ Jeffrey Wu¹ Rewon Child¹ David Luan¹ Dario Amodei^{1*} Ilya Sutskever^{1,2}

Abstract
Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the answers generated by the language model reach 55 F1 on the CoQA dataset – matching or exceeding the performance of 3 out of 4 baseline systems without using the 127,000+ training examples. The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2, is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText. Samples from the model reflect these improvements and contain coherent paragraphs of text. These findings suggest a promising path towards building language processing systems which learn to perform tasks from their naturally occurring demonstrations.

1. Introduction

Machine learning systems now excel (in expectation) at tasks they are trained for by using a combination of large datasets, high-capacity models, and supervised learning (Krizhevsky et al., 2012) (Sutskever et al., 2014) (Amodei et al., 2016). Yet these systems are brittle and sensitive to slight changes in the data distribution (Recht et al., 2018) and task specification (Kirkpatrick et al., 2017). Current systems are better characterized as narrow experts rather than

^{*}Equal contribution. ¹OpenAI, San Francisco, California, United States. Correspondence to: Alec Radford <alec@openai.com>.

competent generalists. We would like to move towards more general systems which can perform many tasks – eventually without the need to manually create and label a training dataset for each one. The dominant approach to creating ML systems is to collect a dataset of training examples demonstrating correct behavior for a desired task, train a system to imitate these behaviors, and then test its performance on independent and identically distributed (IID) held-out examples. This has served well to make progress on narrow experts. But the often erratic behavior of captioning models (Lake et al., 2017), reading comprehension systems (Ito & Lang, 2017), and image classifiers (Alcorn et al., 2018) on the diversity and variety of possible inputs highlights some of the shortcomings of this approach.

Our suspicion is that the prevalence of single task training on single domain datasets is a major contributor to the lack of generalization observed in current systems. Progress towards robust systems with current architectures is likely to require training and measuring performance on a wide range of domains and tasks. Recently, several benchmarks have been proposed such as GLUE (Wang et al., 2018) and decaNLP (McCam et al., 2018) to begin studying this. Multitask learning (Caruana, 1997) is a promising framework for improving general performance. However, multitask training in NLP is still nascent. Recent work reports modest performance improvements (Yogatama et al., 2019) and the two most ambitious efforts to date have trained on a total of 10 and 17 (dataset, subject) pairs respectively (McCam et al., 2018) (Bosman et al., 2018). From a meta-learning perspective, each (dataset, subject) pair is a single training example sampled from the distribution of datasets and objectives. Current ML systems need hundreds to thousands of examples to induce functions which generalize well. This suggests that multitask training may need just as many effective training pairs to realize its promise with current approaches. It will be very difficult to continue to scale the creation of datasets and the design of objectives to the degree that may be required to brute force our way there with current techniques. This motivates exploring additional setups for performing multitask learning.

The current best performing systems on language tasks

Language Models are Few-Shot Learners

Tom R. Brown [*]	Benjamin Mann [*]	Nick Ryder [*]	Melanie Subbiah [*]
Jared Kaplan ¹	Prafulla Dhariwal	Arvind Neelakantan	Pranav Shyam
Amanda Askell	Sandhini Agarwal	Ariel Herbert-Voss	Gretchen Krueger
Rewon Child	Aditya Ramesh	Daniel M. Ziegler	Jeffrey Wu
Christopher Hesse	Mark Chen	Eric Sigler	Mateusz Litwin
Benjamin Chess	Jack Clark	Christopher Berner	
Sam McCandlish	Alec Radford	Ilya Sutskever	Dario Amodei

OpenAI
Abstract

Recent work has demonstrated substantial gains on many NLP tasks and benchmarks by pre-training on a large corpus of text followed by fine-tuning on a specific task. While typically task-agnostic in architecture, this method still requires task-specific fine-tuning datasets of thousands or tens of thousands of examples. By contrast, humans can generally perform a new language task from only a few examples or from simple instructions – something which current NLP systems still largely struggle to do. Here we show that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even reaching competitiveness with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x more than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves strong performance on many NLP datasets, including translation, question-answering, and clone tasks, as well as several tasks that require on-the-fly reasoning or domain adaptation, such as unscrambling words, using a novel word in a sentence, or performing 3-digit arithmetic. At the same time, we also identify some datasets where GPT-3’s few-shot learning still struggles, as well as some datasets where GPT-3 faces methodological issues related to training on large web corpora. Finally, we find that GPT-3 can generate samples of news articles which human evaluators have difficulty distinguishing from articles written by humans. We discuss broader societal impacts of this finding and of GPT-3 in general.

^{*}Equal contribution
¹Johns Hopkins University, OpenAI
Author contributions listed at end of paper.

Common sense reasoning

Question answering

Reading comprehension

Summarization tasks

Simple math and arithmetic

Essay writing

Translation

arXiv:2005.14165v4 [cs.CL] 22 Jul 2020

HellaSwag dataset



A woman is outside with a bucket and a dog. The dog is running around trying to avoid a bath. She...

- A. rinses the bucket off with soap and blow dry the dog's head.
- B. uses a hose to keep it from getting soapy.
- C. gets the dog wet, then it runs away again.**
- D. gets into a bath tub with the dog.



How to determine who has right of way.

Come to a complete halt at a stop sign or red light. At a stop sign, come to a complete halt for about 2 seconds or until vehicles that arrived before you clear the intersection. If you're stopped at a red light, proceed when the light has turned green. ...

- A. Stop for no more than two seconds, or until the light turns yellow. A red light in front of you indicates that you should stop.
- B. After you come to a complete stop, turn off your turn signal. Allow vehicles to move in different directions before moving onto the sidewalk.
- C. Stay out of the oncoming traffic. People coming in from behind may elect to stay left or right.
- D. If the intersection has a white stripe in your lane, stop before this line. Wait until all traffic has cleared before crossing the intersection.**



Winogrande dataset

The Winogrande dataset is a collection of commonsense reasoning challenges designed to evaluate an AI's understanding of language and context. Each challenge typically involves a sentence with a blank, and two possible options for filling in the blank. The task is to choose the option that makes the most sense given the context. Here are some examples of Winogrande dataset challenges:

1. Example 1:

- Sentence: "The trophy doesn't fit into the suitcase because it is too small."
- Options: (A) trophy, (B) suitcase
- Answer: (B) suitcase

2. Example 2:

- Sentence: "Sam tried to paint a picture of the landscape, but he didn't have any ____."
- Options: (A) paint, (B) brushes
- Answer: (A) paint

3. Example 3:

- Sentence: "Jane gave Joan candy because she was very happy."
- Options: (A) Jane, (B) Joan
- Answer: (A) Jane

4. Example 4:

- Sentence: "The book was on the table and fell off when ____."
- Options: (A) the table moved, (B) the book moved
- Answer: (A) the table moved

5. Example 5:

- Sentence: "Alex saw a dog chasing a cat while he was walking."
- Options: (A) dog, (B) Alex
- Answer: (B) Alex

How did GPT3 achieve task-agnostic model training and in-context learning?

Three Breakthrough Ideas

Unsupervised or self-supervised
Learning

Transformer Architecture

Scale

Three Breakthrough Ideas

Unsupervised or self-supervised
Learning

Transformer Architecture

Scale

Treat Internet as source of unlabelled data



And create an algorithmic process that is rich in learning various skills

Pretraining on Internet

Train a transformer model to predict next word on every sentence found on the Internet

For other uses, see [Black hole \(disambiguation\)](#).

A **black hole** is a region of **spacetime** where **gravity** is so strong that nothing, including **light** and other **electromagnetic waves**, has enough energy to escape it.^[2] The theory of **general relativity** predicts that a sufficiently compact **mass** can deform spacetime to form a black hole.^{[3][4]} The **boundary** of no escape is called the **event horizon**. Although it has a great effect on the fate and circumstances of an object crossing it, it has no locally detectable features according to general relativity.^[5] In many ways, a black hole acts like an ideal **black body**, as it reflects no light.^{[6][7]} Moreover, **quantum field theory in curved spacetime** predicts that event horizons emit **Hawking radiation**, with the same spectrum as a black body of a **temperature** inversely proportional to its mass. This temperature is of the order of billionths of a **kelvin** for **stellar black holes**, making it essentially impossible to observe directly.

Objects whose **gravitational fields** are too strong for light to escape were first considered in the 18th century by **John Michell** and **Pierre-Simon Laplace**.^[8] In 1916, **Karl Schwarzschild** found the first modern solution of **general relativity** that would characterize a black hole. **David Finkelstein**, in 1958, first published the interpretation of "black hole" as a region of space from which nothing can escape. Black holes were long considered a mathematical curiosity; it was not until the 1960s that theoretical work showed they were a generic prediction of general relativity. The discovery of **neutron stars** by **Jocelyn Bell Burnell** in 1967 sparked interest in **gravitationally collapsed** compact objects as a possible astrophysical reality. The first black hole known was **Cygnus X-1**, identified by several researchers independently in 1971.^{[9][10]}

Black holes of stellar mass form when massive stars collapse at the end of their life cycle. After a black hole has formed, it can grow by absorbing mass from its surroundings. **Supermassive black holes** of millions of **solar masses** (M_{\odot}) may form by absorbing other stars and merging with other black holes. There is consensus that supermassive black holes exist in the centres of most **galaxies**.

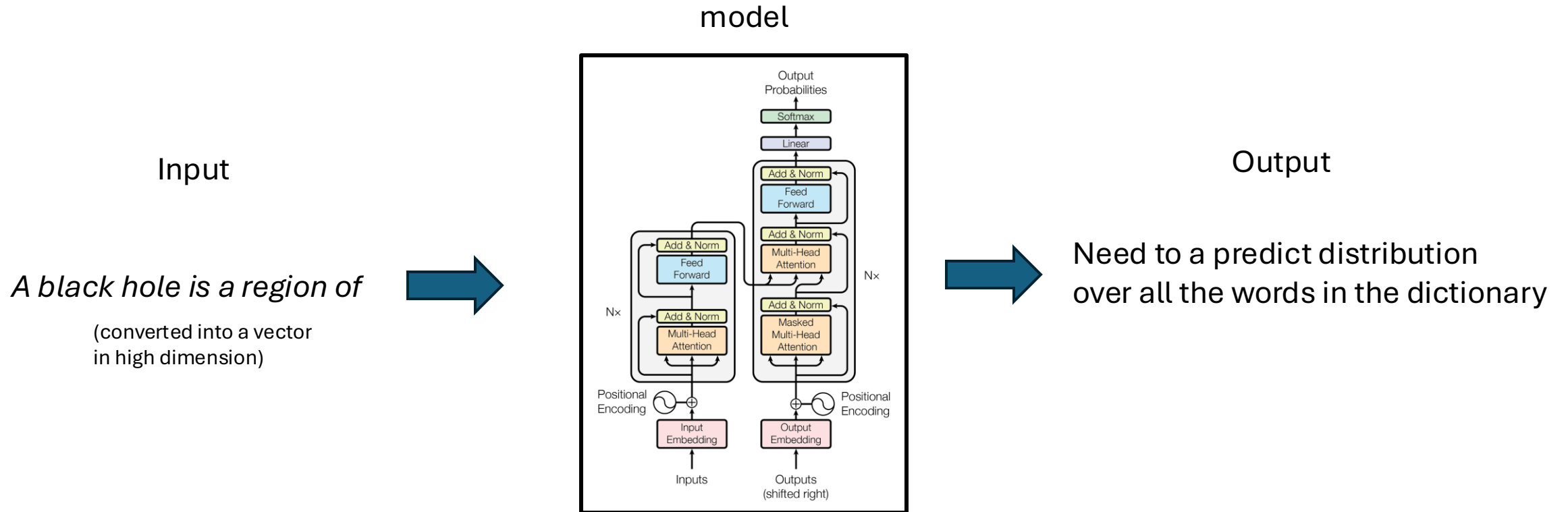
The presence of a black hole can be inferred through its interaction with other **matter** and with electromagnetic radiation such as visible light. Any matter that falls onto a black hole can form an external **accretion disk** heated by **friction**, forming **quasars**, some of the brightest objects in the universe. Stars passing too close to a supermassive black hole can be shredded into streamers that shine very brightly before being "swallowed."^[11] If other stars are orbiting a black hole, their orbits can be used to determine the black hole's mass and location. Such observations can be used to exclude possible alternatives such as neutron stars. In this way, astronomers have identified numerous stellar black hole candidates in **binary systems** and established that the radio source known as **Sagittarius A***, at the core of the **Milky Way** galaxy, contains a supermassive black hole of about 4.3 million **solar masses**.



A black hole is a region of -----

general relativity that would ----

What is Input and Output?



Similar to handwritten recognition task we saw earlier
We usually train them so that loss goes to zero on the training data

Pretraining on Internet: Keys to Success

Train a transformer model to predict next word on every sentence found on the Internet

universal DNN model

2 unsupervised

4 rich representation of languages and reasoning

1 no need to create Labelled datasets.

3 Unlimited dataset
Remember: DNNs needs lots of data and compute!

Language Models are Unsupervised Multitask Learners

Alec Radford¹, Jeffrey Wu¹, Rewon Child¹, David Luan¹, Dario Amodei^{1,2}, Rysa Sutskever^{1,2}

Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the answers generated by the language model reach 55 F1 on the CoQA dataset - matching or exceeding the performance of 7 out of 8 baseline systems without using the 127,000+ training examples. The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2, is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText. Samples from the model reflect these improvements and contain coherent paragraphs of text. These findings suggest a promising path towards building language processing systems which learn to perform tasks from their naturally occurring demonstrations.

1. Introduction

Machine learning systems now excel (in expectation) at tasks they are trained for by using a combination of large datasets, high-capacity models, and supervised learning (Kotlovsky et al., 2012) (Sutskever et al., 2014) (Amodei et al., 2016). Yet these systems are brittle and sensitive to slight changes in the data distribution (Blech et al., 2018) and task specification (Kirkpatrick et al., 2017). Current systems are better characterized as narrow experts rather than

competent generalists. We would like to move towards more general systems which can perform many tasks - eventually without the need to manually create and label a training dataset for each one.

The dominant approach to creating ML systems is to collect a dataset of training examples demonstrating correct behavior for a desired task, train a system to imitate these behaviors, and then test its performance on independent and identically distributed (IID) held-out examples. This has served well to make progress on narrow experts. But the often erratic behavior of captioning models (Lake et al., 2017), reading comprehension systems (Jia & Luong, 2017), and image classifiers (Alison et al., 2018) on the diversity and variety of possible inputs highlights some of the shortcomings of this approach.

Our suspicion is that the prevalence of single task training on single domain datasets is a major contributor to the lack of generalization observed in current systems. Progress towards robust systems with current architectures is likely to require training and measuring performance on a wide range of domains and tasks. Recently, several benchmarks have been proposed such as GLUE (Wang et al., 2018) and SuperGLUE (McCann et al., 2018) to begin studying this.

Multitask learning (Caruana, 1997) is a promising framework for improving general performance. However, multitask training in NLP is still nascent. Recent work reports modest performance improvements (Vigilantini et al., 2019) and the two most ambitious efforts to date have trained on a total of 10 and 17 (data.enr., 2019) pairs respectively (McCann et al., 2018) (Bowman et al., 2018). From a meta-learning perspective, each (dataset, objective) pair is a single training example sampled from the distribution of datasets and objectives. Current ML systems need hundreds to thousands of examples to induce functions which generalize well. This suggests that multitask training may need just as many effective training pairs to realize its promise with current approaches. It will be very difficult to continue to scale the creation of datasets and the design of objectives to the degree that may be required to brute force our way there with current techniques. This motivates exploring additional setups for performing multitask learning.

The current best performing systems on language tasks

¹Equal contribution. ²OpenAI, San Francisco, California, United States. Correspondence to: Alec Radford <alec@openai.com>.

Why next word prediction has rich structure?

Suppose GPT3 is going over the paper by Thomas

0107 ADP 2010
[CS.DM] 17 Apr 2010
/4

Constructive discrepancy minimization for convex sets

Thomas Rothvot*
University of Washington, Seattle

Abstract

A classical theorem of Spencer shows that any set system with n sets and n elements admits a coloring of discrepancy $O(\sqrt{n})$. Recent exciting work of Bansal, Lovett and Meka shows that such colorings can be found in polynomial time. In fact, the Lovett-Meka algorithm finds a half integral point in any “large enough” polytope. However, their algorithm crucially relies on the fact structure and does not apply to general convex sets.

We show that for any symmetric convex set K with Gaussian measure at least $e^{-n/10}$, the following algorithm finds a point $y \in K \cap [-1, 1]^n$ with $\|y\|_2$ coordinates in ± 1 : (1) take a random Gaussian vector x ; (2) compute the point y in $K \cap [-1, 1]^n$ that is closest to x ; (3) return y .

This provides another truly constructive proof of Spencer’s theorem and the first constructive proof of a Theorem of Gluskin and Giannopoulos.

and it encounters this

Lemma 6. *Let $P, Q \subseteq \mathbb{R}^n$ be convex sets and let $g : \mathbb{R}^n \rightarrow \mathbb{R}$ be a strictly convex function. Suppose that x^* is an optimum solution to $\min\{g(x) \mid x \in P \cap Q\}$ and x^* lies in the interior of Q . Then x^* is also an optimum solution to $\min\{g(x) \mid x \in P\}$.*

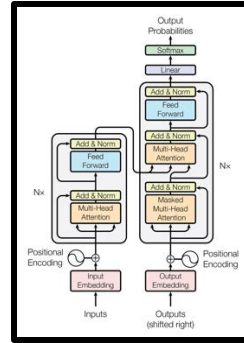
Proof. Suppose for the sake of contradiction that there is a $y^* \in P$ with $g(y^*) < g(x^*)$, then some convex combination $(1 - \lambda)y^* + \lambda x^*$ with $0 < \lambda < 1$ lies also in Q and has a better objective function than x^* , which is a ~~contradiction~~. \square



The model needs to predict that the missing word is “contradiction”

How do we use such a network (Inference)?

A sequence of words



Predict the next word

Use the model Iteratively!!

Autoregressive

=

The model consumes previously generated output

Let's say I want the model to complete a sentence:
I love playing.



I love playing → tennis

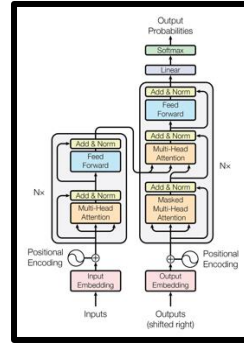
I love playing tennis → Roger

I love playing tennis. Roger → is

I love playing tennis. Roger is → my

How do we use such a network (Inference)?

A sequence of words



Predict the next word

Use the model iteratively

Autoregressive

assumes

previously generated output

Can benefit from online/dynamic algorithmic ideas

Let the model
complete a sentence:
I love playing.



I love playing → tennis

I love playing tennis → Roger

I love playing tennis. Roger → is

I love playing tennis. Roger is → my

“Prompt” the model (in-context learning)

Problem: Beth bakes 4, 2 dozen batches of cookies in a week. If these cookies are shared amongst 16 people equally, how many cookies does each person consume?

Solution: Beth bakes 4 2 dozen batches of cookies for a total of $4 \times 2 = \ll 4 \times 2 = 8 \gg$ 8 dozen cookies
There are 12 cookies in a dozen and she makes 8 dozen cookies for a total of $12 \times 8 = \ll 12 \times 8 = 96 \gg$ 96 cookies
She splits the 96 cookies equally amongst 16 people so they each eat $96/16 = \ll 96/16 = 6 \gg$ 6 cookies

Final Answer: 6

Problem: Mrs. Lim milks her cows twice a day. Yesterday morning, she got 68 gallons of milk and in the evening, she got 82 gallons. This morning, she got 18 gallons fewer than she had yesterday morning. After selling some gallons of milk in the afternoon, Mrs. Lim has only 24 gallons left. How much was her revenue for the milk if each gallon costs \$3.50?

Mrs. Lim got 68 gallons - 18 gallons = $\ll 68 - 18 = 50 \gg$ 50 gallons this morning.
So she was able to get a total of 68 gallons + 82 gallons + 50 gallons = $\ll 68 + 82 + 50 = 200 \gg$ 200 gallons.
She was able to sell 200 gallons - 24 gallons = $\ll 200 - 24 = 176 \gg$ 176 gallons.
Thus, her total revenue for the milk is \$3.50/gallon x 176 gallons = $\$ \ll 3.50 \times 176 = 616 \gg$ 616.

Final Answer: 616

Problem: Tina buys 3 12-packs of soda for a party. Including Tina, 6 people are at the party. Half of the people at the party have 3 sodas each, 2 of the people have 4, and 1 person has 5. How many sodas are left over when the party is over?

Solution: Tina buys 3 12-packs of soda, for $3 \times 12 = \ll 3 \times 12 = 36 \gg$ 36 sodas
6 people attend the party, so half of them is $6/2 = \ll 6/2 = 3 \gg$ 3 people
Each of those people drinks 3 sodas, so they drink $3 \times 3 = \ll 3 \times 3 = 9 \gg$ 9 sodas
Two people drink 4 sodas, which means they drink $2 \times 4 = \ll 4 \times 2 = 8 \gg$ 8 sodas
With one person drinking 5, that brings the total drank to $5 + 9 + 8 + 3 = \ll 5 + 9 + 8 + 3 = 25 \gg$ 25 sodas
As Tina started off with 36 sodas, that means there are $36 - 25 = \ll 36 - 25 = 11 \gg$ 11 sodas left

Final Answer: 11

Pretraining on Internet

Train a transformer model to predict next word on every sentence found on the Internet

universal DNN model

2 unsupervised

4 rich representation of languages

1

no need to create
Labelled datasets.

3

Unlimited dataset
Remember: DNNs needs lots of
data and compute!

Language Models are Unsupervised Multitask Learners

Alec Radford¹, Jeffrey Wu¹, Rewon Child¹, David Luan¹, Dario Amodei^{1,2}, Rya Sutskever^{1,2}

Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the answers generated by the language model reach 55 F1 on the CoQA dataset - matching or exceeding the performance of 3 out of 4 baseline systems without using the 127,000+ training examples. The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2, is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText. Samples from the model reflect these improvements and contain coherent paragraphs of text. These findings suggest a promising path towards building language processing systems which learn to perform tasks from their naturally occurring demonstrations.

1. Introduction

Machine learning systems now excel (in expectation) at tasks they are trained for by using a combination of large datasets, high-capacity models, and supervised learning (Kotlovsky et al., 2012) (Sutskever et al., 2014) (Amodei et al., 2016). Yet these systems are brittle and sensitive to slight changes in the data distribution (Blech et al., 2018) and task specification (Kirkpatrick et al., 2017). Current systems are better characterized as narrow experts rather than

competent generalists. We would like to move towards more general systems which can perform many tasks - eventually without the need to manually create and label a training dataset for each one.

The dominant approach to creating ML systems is to collect a dataset of training examples demonstrating correct behavior for a desired task, train a system to imitate these behaviors, and then test its performance on independent and identically distributed (IID) held-out examples. This has served well to make progress on narrow experts. But the often erratic behavior of captioning models (Lake et al., 2017), reading comprehension systems (Jia & Luong, 2017), and image classifiers (Alison et al., 2018) on the diversity and variety of possible inputs highlights some of the shortcomings of this approach.

Our suspicion is that the prevalence of single task training on single domain datasets is a major contributor to the lack of generalization observed in current systems. Progress towards robust systems with current architectures is likely to require training and measuring performance on a wide range of domains and tasks. Recently, several benchmarks have been proposed such as GLUE (Wang et al., 2018) and SuperGLUE (McCann et al., 2018) to begin studying this.

Multitask learning (Caruana, 1997) is a promising framework for improving general performance. However, multitask training in NLP is still nascent. Recent work reports modest performance improvements (Vigilantini et al., 2019) and the two most ambitious efforts to date have trained on a total of 10 and 17 (data.enr., 2019) (data.enr., 2019) pairs respectively (McCann et al., 2018) (Bowman et al., 2018). From a meta-learning perspective, each (dataset, objective) pair is a single training example sampled from the distribution of datasets and objectives. Current ML systems need hundreds to thousands of examples to induce functions which generalize well. This suggests that multitask training may need just as many effective training pairs to realize its promise with current approaches. It will be very difficult to continue to scale the creation of datasets and the design of objectives to the degree that may be required to brute force our way there with current techniques. This motivates exploring additional setups for performing multitask learning.

The current best performing systems on language tasks

¹Equal contribution. ²OpenAI, San Francisco, California, United States. Correspondence to: Alec Radford <alec@openai.com>.

Pretraining on Internet

Train a transformer model to predict next word on every sentence found on the Internet

universal DNN model

2 unsupervised

4 rich representation of languages

1

no need to create
Labelled

Is this specific to Language?
Can we reproduce in other domains?

3

Unlimited dataset
Remember: DNNs needs lots of
data and compute!

Language Models are Unsupervised Multitask Learners

Alec Radford¹, Jeffrey Wu¹, Rewon Child¹, David Luan¹, Dario Amodei²

Abstract

Natural language processing tasks such as machine translation, text summarization, question answering, machine reading comprehension, and many others have been successfully addressed by deep learning models. However, these models are trained on a single task and are therefore not able to generalize to other tasks. In this paper, we propose a new framework for training language models that allows them to learn from a wide range of tasks simultaneously. We call this framework "Language Models are Unsupervised Multitask Learners".

Our suspicion is that the prevalence of single task training on single domain datasets is a major contributor to the lack of generalization observed in current systems. Progress towards robust systems with current architectures is likely to require training and measuring performance on a wide range of domains and tasks. Recently, several benchmarks have been proposed such as GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019) to begin studying this.

Multitask learning (Caruana, 1997) is a promising framework for improving general performance. However, multitask training in NLP is still nascent. Recent work reports modest performance improvements (Vigilantini et al., 2019) and the two most ambitious efforts to date have trained on a total of 10 and 17 (data sets, object levels) pairs respectively (McCann et al., 2018) (Bowman et al., 2018). From a meta-learning perspective, each (data set, object level) pair is a single training example sampled from the distribution of datasets and objectives. Current ML systems need hundreds to thousands of examples to induce functions which generalize well. This suggests that multitask training may need just as many effective training pairs to realize its promise with current approaches. It will be very difficult to continue to scale the creation of datasets and the design of objectives to the degree that may be required to brute force our way through current techniques. This motivates exploring additional setups for performing multitask learning.

The current best performing systems on language tasks

1. Introduction

Machine learning systems now excel (in expectation) at tasks they are trained for by using a combination of large datasets, high-capacity models, and supervised learning (Kotlyevsky et al., 2012) (Sutskever et al., 2014) (Anseli et al., 2016). Yet these systems are brittle and sensitive to slight changes in the data distribution (Bleher et al., 2018) and task specification (Kirkpatrick et al., 2017). Current systems are better characterized as narrow experts rather than generalists.

¹Equal contribution. ²OpenAI, San Francisco, California, United States. Correspondence to: Alec Radford <alec@openai.com>.

Can we do this for images?

Natural Idea: Predict the next pixel.

Almost works as seen from Image GPT paper. Expensive.



Research ▾ API ▾ ChatGPT ▾ Safety Company ▾

Research

Image GPT

We find that, just as a large transformer model trained on language can generate coherent text, the same exact model trained on pixel sequences can generate coherent image completions and samples. By establishing a correlation between sample quality and image classification accuracy, we show that our best generative model also contains features competitive with top convolutional nets in the unsupervised setting.

https://cdn.openai.com/papers/Generative_Pretraining_from_Pixels_V2.pdf

Generative Pretraining from Pixels

Mark Chen¹ Alec Radford¹ Rewon Child¹ Jeff Wu¹ Heewoo Jun¹ David Luan¹ Ilya Sutskever¹

Abstract

Inspired by progress in unsupervised representation learning for natural language, we examine whether similar models can learn useful representations for images. We train a sequence Transformer to auto-regressively predict pixels, without incorporating knowledge of the 2D input structure. Despite training on low-resolution ImageNet without labels, we find that a GPT-2 scale model learns strong image representations as measured by linear probing, fine-tuning, and low-data classification. On CIFAR-10, we achieve 96.3% accuracy with a linear probe, outperforming a supervised Wide ResNet, and 99.0% accuracy with full fine-tuning, matching the top supervised pre-trained models. We are also competitive with self-supervised benchmarks on ImageNet when substituting pixels for a VQVAE encoding, achieving 69.0% top-1 accuracy on a linear probe of our features.

1. Introduction

Unsupervised pre-training played a central role in the resurgence of deep learning. Starting in the mid 2000's, approaches such as the Deep Belief Network (Hinton et al., 2006) and Denoising Autoencoder (Vincent et al., 2008) were commonly used in neural networks for computer vision (Lee et al., 2009) and speech recognition (Mohamed et al., 2009). It was believed that a model which learned the data distribution $P(X)$ would also learn beneficial features for the subsequent supervised modeling of $P(Y|X)$ (Lasserre et al., 2006; Erhan et al., 2010). However, advancements such as piecewise linear activation functions (Nair & Hinton, 2010), improved initializations (Glorot & Bengio, 2010), and normalization strategies (Ioffe & Szegedy, 2015; Ba et al., 2016) removed the need for pre-training in order to achieve strong results. Other research cast doubt

on the benefits of *deep* unsupervised representations and reported strong results using a single layer of learned features (Coates et al., 2011), or even random features (Huang et al., 2014; May et al., 2017). The approach fell out of favor as the state of the art increasingly relied on directly encoding prior structure into the model and utilizing abundant supervised data to directly learn representations (Krizhevsky et al., 2012; Graves & Jaitly, 2014). Retrospective study of unsupervised pre-training demonstrated that it could even hurt performance in modern settings (Paine et al., 2014).

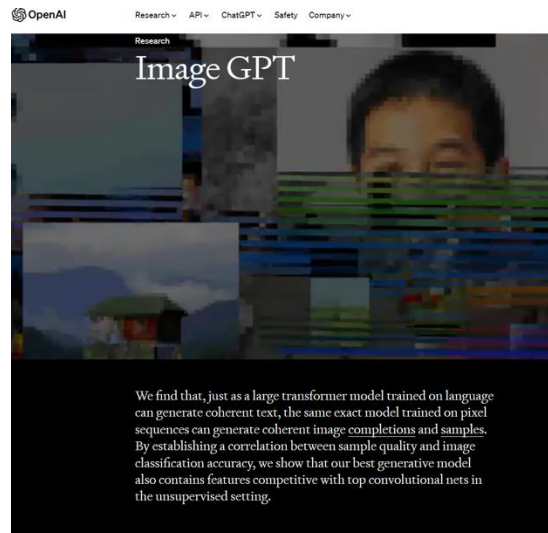
Instead, unsupervised pre-training flourished in a different domain. After initial strong results for word vectors (Mikolov et al., 2013), it has pushed the state of the art forward in Natural Language Processing on most tasks (Dai & Le, 2015; Peters et al., 2018; Howard & Ruder, 2018; Radford et al., 2018; Devlin et al., 2018). Interestingly, the training objective of a dominant approach like BERT, the prediction of corrupted inputs, closely resembles that of the Denoising Autoencoder, which was originally developed for images.

As a higher dimensional, noisier, and more redundant modality than text, images are believed to be difficult for generative modeling. Here, self-supervised approaches designed to encourage the modeling of more global structure (Doersch et al., 2015) have shown significant promise. A combination of new training objectives (Oord et al., 2018), more recent architectures (Gomez et al., 2017), and increased model capacity (Kolesnikov et al., 2019) has allowed these methods to achieve state of the art performance in low data settings (Hénaff et al., 2019) and sometimes even outperform supervised representations in transfer learning settings (He et al., 2019; Misra & van der Maaten, 2019).

Given that it has been a decade since the original wave of generative pre-training methods for images and considering their substantial impact in NLP, this class of methods is due for a modern re-examination and comparison with the recent progress of self-supervised methods. We re-evaluate generative pre-training on images and demonstrate that when using a flexible architecture (Vaswani et al., 2017), a tractable and efficient likelihood based training objective (Larochelle & Murray, 2011; Oord et al., 2016), and significant compute resources (1024 TPU cores), generative pre-training is competitive with other self-supervised approaches and learns

¹Equal contribution ¹OpenAI, San Francisco, CA, USA. Correspondence to: Mark Chen <mark@openai.com>.

How would you use this network to do digit recognition? *Finetuning*



Learn a linear layer using
few samples



Softmax to get a
distribution over
the output space

Finetuning: a general concept where you take a pretrained model and train further on the downstream task of interest. Cheaper than pretraining, and helps to improve performance over incontext learning sometimes.

Natural Idea: Predict the next pixel.

Almost works as seen from Image GPT paper. Expensive.



Research ▾ API ▾ ChatGPT ▾ Safety Company ▾

Research

Image GPT

We find that, just as a large transformer model trained on language can generate coherent text, the same exact model trained on pixel sequences can generate coherent image completions and samples. By establishing a correlation between sample quality and image classification accuracy, we show that our best generative model also contains features competitive with top convolutional nets in the unsupervised setting.

https://cdn.openai.com/papers/Generative_Pretraining_from_Pixels_V2.pdf

Generative Pretraining from Pixels

Mark Chen¹ Alec Radford¹ Rewon Child¹ Jeff Wu¹ Heewoo Jun¹ David Luan¹ Ilya Sutskever¹

Abstract

Inspired by progress in unsupervised representation learning for natural language, we examine whether similar models can learn useful representations for images. We train a sequence Transformer to auto-regressively predict pixels, without incorporating knowledge of the 2D input structure. Despite training on low-resolution ImageNet without labels, we find that a GPT-2 scale model learns strong image representations as measured by linear probing, fine-tuning, and low-data classification. On CIFAR-10, we achieve 96.3% accuracy with a linear probe, outperforming a supervised Wide ResNet, and 99.0% accuracy with full fine-tuning, matching the top supervised pre-trained models. We are also competitive with self-supervised benchmarks on ImageNet when substituting pixels for a VQVAE encoding, achieving 69.0% top-1 accuracy on a linear probe of our features.

1. Introduction

Unsupervised pre-training played a central role in the resurgence of deep learning. Starting in the mid 2000's, approaches such as the Deep Belief Network (Hinton et al., 2006) and Denoising Autoencoder (Vincent et al., 2008) were commonly used in neural networks for computer vision (Lee et al., 2009) and speech recognition (Mohamed et al., 2009). It was believed that a model which learned the data distribution $P(X)$ would also learn beneficial features for the subsequent supervised modeling of $P(Y|X)$ (Lasserre et al., 2006; Erhan et al., 2010). However, advancements such as piecewise linear activation functions (Nair & Hinton, 2010), improved initializations (Glorot & Bengio, 2010), and normalization strategies (Ioffe & Szegedy, 2015; Ba et al., 2016) removed the need for pre-training in order to achieve strong results. Other research cast doubt

on the benefits of *deep* unsupervised representations and reported strong results using a single layer of learned features (Coates et al., 2011), or even random features (Huang et al., 2014; May et al., 2017). The approach fell out of favor as the state of the art increasingly relied on directly encoding prior structure into the model and utilizing abundant supervised data to directly learn representations (Krizhevsky et al., 2012; Graves & Jaitly, 2014). Retrospective study of unsupervised pre-training demonstrated that it could even hurt performance in modern settings (Paine et al., 2014).

Instead, unsupervised pre-training flourished in a different domain. After initial strong results for word vectors (Mikolov et al., 2013), it has pushed the state of the art forward in Natural Language Processing on most tasks (Dai & Le, 2015; Peters et al., 2018; Howard & Ruder, 2018; Radford et al., 2018; Devlin et al., 2018). Interestingly, the training objective of a dominant approach like BERT, the prediction of corrupted inputs, closely resembles that of the Denoising Autoencoder, which was originally developed for images.

As a higher dimensional, noisier, and more redundant modality than text, images are believed to be difficult for generative modeling. Here, self-supervised approaches designed to encourage the modeling of more global structure (Doersch et al., 2015) have shown significant promise. A combination of new training objectives (Oord et al., 2018), more recent architectures (Gomez et al., 2017), and increased model capacity (Kolesnikov et al., 2019) has allowed these methods to achieve state of the art performance in low data settings (Hénaff et al., 2019) and sometimes even outperform supervised representations in transfer learning settings (He et al., 2019; Misra & van der Maaten, 2019).

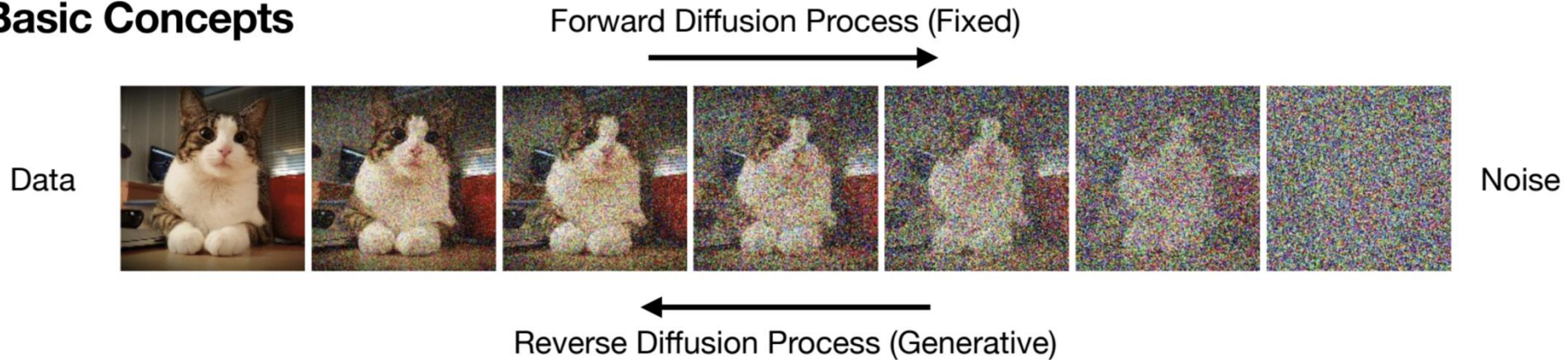
Given that it has been a decade since the original wave of generative pre-training methods for images and considering their substantial impact in NLP, this class of methods is due for a modern re-examination and comparison with the recent progress of self-supervised methods. We re-evaluate generative pre-training on images and demonstrate that when using a flexible architecture (Vaswani et al., 2017), a tractable and efficient likelihood based training objective (Larochelle & Murray, 2011; Oord et al., 2016), and significant compute resources (1024 TPU cores), generative pre-training is competitive with other self-supervised approaches and learns

¹Equal contribution ¹OpenAI, San Francisco, CA, USA. Correspondence to: Mark Chen <mark@openai.com>.

Diffusion Models (Ho et al)

Denoising Diffusion Probabilistic Models (DDPM)

Basic Concepts

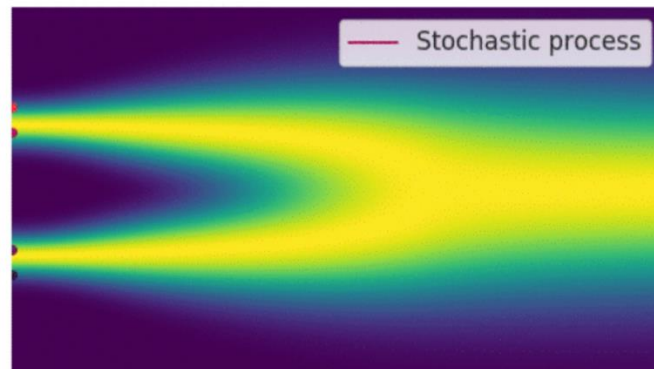


DDPM consists of two processes:

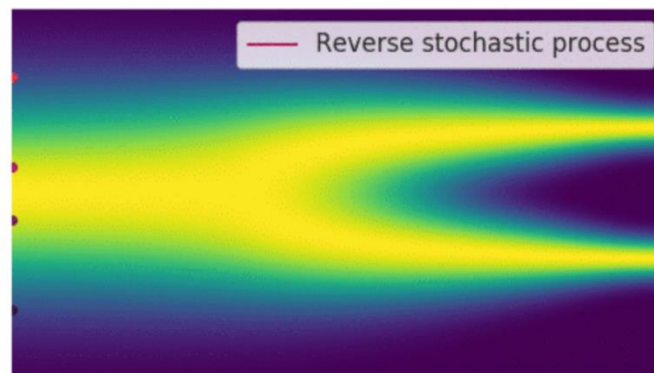
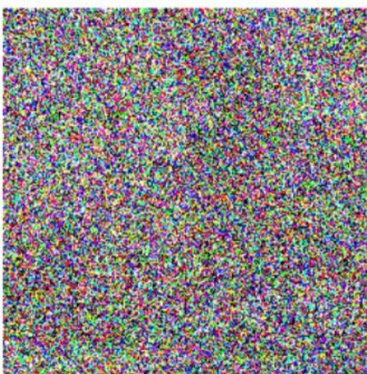
1. Forward diffusion process gradually adds noise to the input
2. Reverse denoising process learns to generate data by denoising

Denoising Diffusion Probabilistic Models (DDPM)

Basic Concepts



Forward Process:
“Destroy” data by gradually adding
small amounts of Gaussian noise

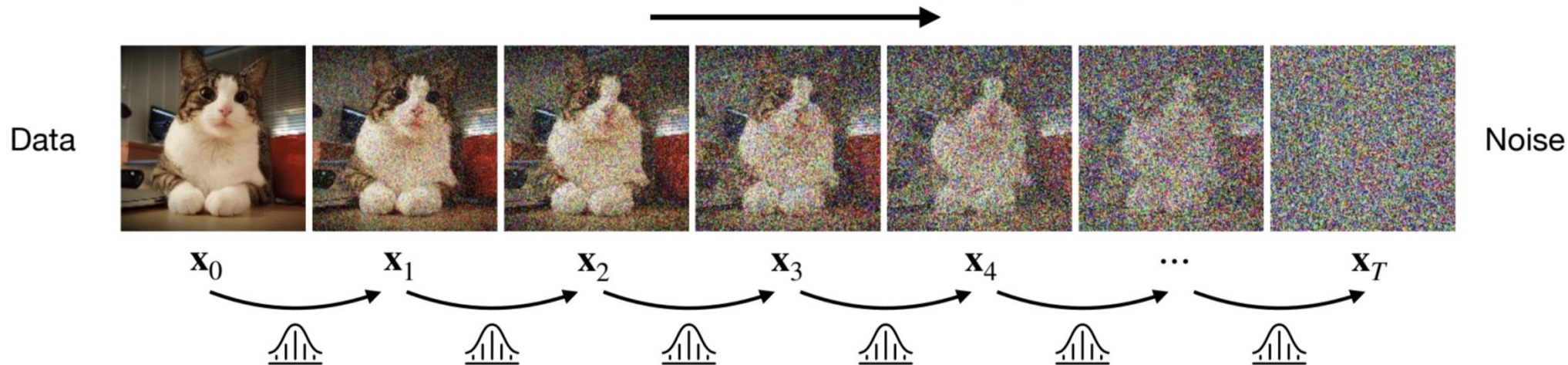


Reverse Process:
“Create” data by gradually denoising a
noisy code from a stationary distribution

Denoising Diffusion Probabilistic Models (DDPM)

Forward Process

Forward Diffusion Process (Fixed)



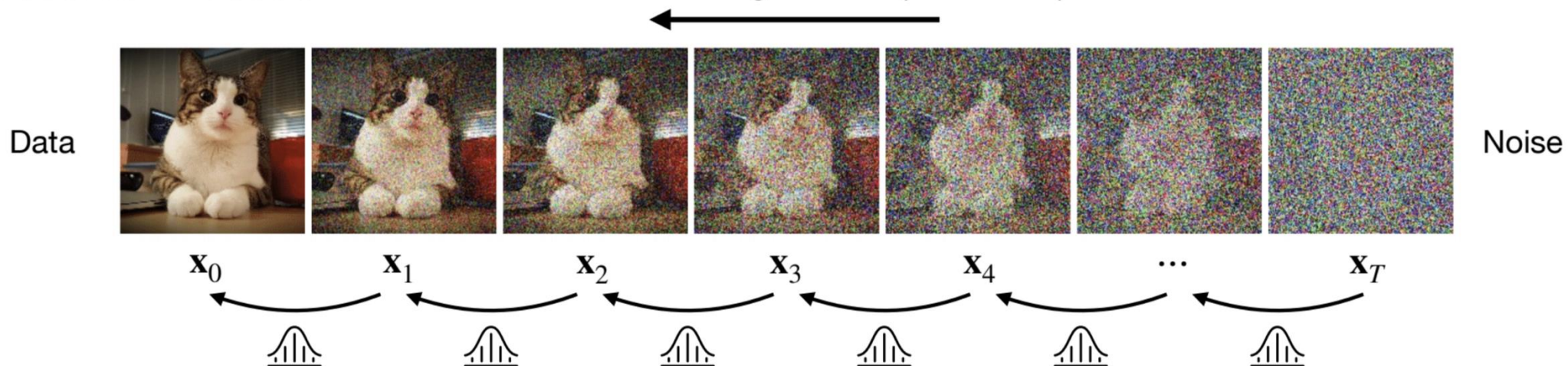
$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \overset{\text{Mean}}{\sqrt{1 - \beta_t} \mathbf{x}_{t-1}}, \overset{\text{Variance}}{\beta_t \mathbf{I}})$$

Gaussian Distribution Variance

Denoising Diffusion Probabilistic Models (DDPM)

Reverse Process

Reverse Denoising Process (Generative)

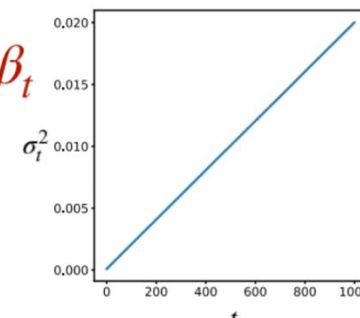


When β_t is small, $q(\mathbf{x}_{t-1} | \mathbf{x}_t)$ is also a Gaussian.

$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad p(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t; \mu_\theta(\mathbf{x}_t), \sigma_t^2 \mathbf{I})$$

Learnable

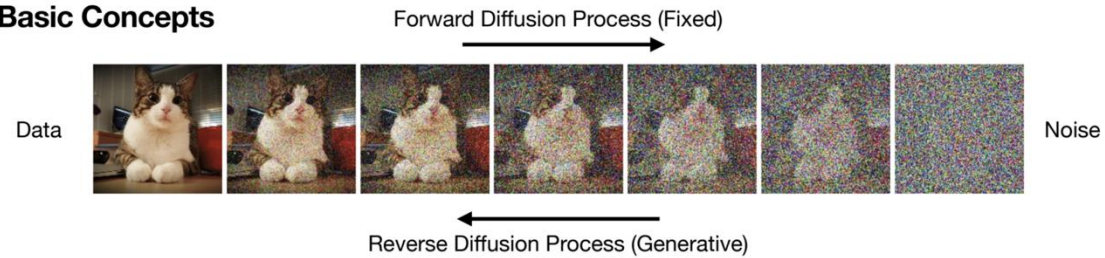
E.g., $\sigma_t^2 = \beta_t$



What are general principles?

Diffusion process

Basic Concepts



- No need for supervision
 - No need for labels
 - Rich and unlimited datasets on Internet
 - Use DNNs to learn!
-

DINO Process

Idea: give different **views** of the a single picture and try to learn to reconstruct the full picture.

Emerging Properties in Self-Supervised Vision Transformers

Mathilde Caron^{1,2} Hugo Touvron^{1,3} Ishan Misra¹ Hervé Jegou¹
Julien Mairal² Piotr Bojanowski¹ Armand Joulin¹

¹ Facebook AI Research ² Inria* ³ Sorbonne University



Figure 1: **Self-attention from a Vision Transformer with 8×8 patches trained with no supervision.** We look at the self-attention of the [CLS] token on the heads of the last layer. This token is not attached to any label nor supervision. These maps show that the model automatically learns class-specific features leading to unsupervised object segmentations.

Abstract

*In this paper, we question if self-supervised learning provides new properties to Vision Transformer (ViT) [19] that stand out compared to convolutional networks (convnets). Beyond the fact that adapting self-supervised methods to this architecture works particularly well, we make the following observations: first, self-supervised ViT features contain explicit information about the semantic segmentation of an image, which does not emerge as clearly with supervised ViTs, nor with convnets. Second, these features are also excellent k-NN classifiers, reaching 78.3% top-1 on ImageNet with a small ViT. Our study also underlines the importance of momentum encoder [33], multi-crop training [10], and the use of small patches with ViTs. We implement our findings into a simple self-supervised method, called DINO, which we interpret as a form of self-distillation with **no** labels. We show the synergy between DINO and ViTs by achieving 80.1% top-1 on ImageNet in linear evaluation with ViT-Base.*

1. Introduction

Transformers [70] have recently emerged as an alternative to convolutional neural networks (convnets) for visual recognition [19, 69, 83]. Their adoption has been coupled with a training strategy inspired by natural language processing (NLP), that is, pretraining on large quantities of data and finetuning on the target dataset [18, 55]. The resulting Vision Transformers (ViT) [19] are competitive with convnets but, they have not yet delivered clear benefits over them: they are computationally more demanding, require more training data, and their features do not exhibit unique properties.

In this paper, we question whether the muted success of Transformers in vision can be explained by the use of supervision in their pretraining. Our motivation is that one of the main ingredients for the success of Transformers in NLP was the use of self-supervised pretraining, in the form of close procedure in BERT [18] or language modeling in GPT [55]. These self-supervised pretraining objectives use the words in a sentence to create pretext tasks that provide a richer learning signal than the supervised objective of predicting a single label per sentence. Similarly, in images, image-level supervision often reduces the rich visual information contained in an image to a single concept selected from a predefined set of a few thousand categories of objects [60].

While the self-supervised pretext tasks used in NLP are

*Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France.
Correspondence: mathilde@fb.com
Code: <https://github.com/facebookresearch/dino>

Can we pretrain a model that can recognize objects/describe images in natural language without explicitly training on it?

And reverse it? That is, given text description generate an image?

CLIP Paper (Learning Transferable Visual Models From Natural Language Supervision, *Radford et al*)

Search all images on the Internet and captions of those images.

Learn a model that maps images and the text surrounding it to **same space**!

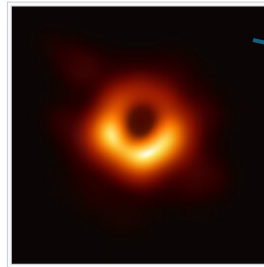
For other uses, see [Black hole \(disambiguation\)](#).

A **black hole** is a region of [spacetime](#) where [gravity](#) is so strong that nothing, including [light](#) and other [electromagnetic waves](#), has enough energy to escape it.^[2] The theory of [general relativity](#) predicts that a sufficiently compact [mass](#) can deform spacetime to form a black hole.^{[3][4]} The [boundary](#) of no escape is called the [event horizon](#). Although it has a great effect on the fate and circumstances of an object crossing it, it has no locally detectable features according to general relativity.^[5] In many ways, a black hole acts like an ideal [black body](#), as it reflects no light.^{[6][7]} Moreover, [quantum field theory in curved spacetime](#) predicts that event horizons emit [Hawking radiation](#), with the same spectrum as a black body of a temperature inversely proportional to its mass. This temperature is of the order of billionths of a [kelvin](#) for [stellar black holes](#), making it essentially impossible to observe directly.

Objects whose [gravitational fields](#) are too strong for light to escape were first considered in the 18th century by [John Michell](#) and [Pierre-Simon Laplace](#).^[8] In 1916, [Karl Schwarzschild](#) found the first modern solution of general relativity that would characterize a black hole. [David Finkelstein](#), in 1958, first published the interpretation of "black hole" as a region of space from which nothing can escape. Black holes were long considered a mathematical curiosity; it was not until the 1960s that theoretical work showed they were a generic prediction of general relativity. The discovery of [neutron stars](#) by [Jocelyn Bell Burnell](#) in 1967 sparked interest in [gravitationally collapsed](#) compact objects as a possible astrophysical reality. The first black hole known was [Cygnus X-1](#), identified by several researchers independently in 1971.^{[9][10]}

Black holes of stellar mass form when massive stars collapse at the end of their life cycle. After a black hole has formed, it can grow by absorbing mass from its surroundings. [Supermassive black holes](#) of millions of [solar masses](#) (M_{\odot}) may form by absorbing other stars and merging with other black holes. There is consensus that supermassive black holes exist in the centres of most [galaxies](#).

The presence of a black hole can be inferred through its interaction with other [matter](#) and with electromagnetic radiation such as visible light. Any matter that falls onto a black hole can form an external [accretion disk](#) heated by [friction](#), forming [quasars](#), some of the brightest objects in the universe. Stars passing too close to a supermassive black hole can be shredded into streamers that shine very brightly before being "swallowed."^[11] If other stars are orbiting a black hole, their orbits can be used to determine the black hole's mass and location. Such observations can be used to exclude possible alternatives such as neutron stars. In this way, astronomers have identified numerous stellar black hole candidates in [binary systems](#) and established that the radio source known as [Sagittarius A*](#), at the core of the [Milky Way](#) galaxy, contains a supermassive black hole of about 4.3 million [solar masses](#).



Direct radio image of a supermassive black hole at the core of Messier 87^[1]



Animated simulation of a Schwarzschild black hole with a galaxy passing behind. Around the time of alignment, extreme gravitational lensing of the galaxy is observed.

Image Embedding

Text Embedding

Should be close!

Image Embedding

Text Embedding

Should be close!

Intuition: Concepts in images and the text should be the same!

"Halle Berry" neuron

nature

Explore content ▾

About the journal ▾


Publish with us ▾

Subscribe

[nature](#) > [letters](#) > article

Letter | Published: 23 June 2005

Invariant visual representation by single neurons in the human brain

[R. Quiñan Quiroga](#) , [L. Reddy](#), [G. Kreiman](#), [C. Koch](#) & [J. Fried](#)

[Nature](#) **435**, 1102–1107 (2005) | [Cite this article](#)

52k Accesses | **1121** Citations | **402** Altmetric | [Metrics](#)

Abstract

It takes a fraction of a second to recognize a person or an object even when seen under strikingly different conditions. How such a robust, high-level representation is achieved by neurons in the human brain is still unclear^{[1,2,3,4,5,6](#)}. In monkeys, neurons in the upper stages of the ventral visual pathway respond to complex images such as faces and objects and show some degree of invariance to metric properties such as the stimulus size, position and viewing angle^{[2,4,7,8,9,10,11,12](#)}. We have previously shown that neurons in the human medial temporal lobe (MTL) fire selectively to images of faces, animals, objects or scenes^{[13,14](#)}. Here we report on a remarkable subset of MTL neurons that are selectively activated by strikingly different pictures of given individuals, landmarks or objects and in some cases even by letter strings with their names. These results suggest an invariant, sparse and explicit code, which might be important in the transformation of complex visual percepts into long-term and more abstract memories.

CLIP Paper (Learning Transferable Visual Models From Natural Language Supervision)

Search all images on the Internet and captions of those images.

Learn a model that maps images and the text surrounding it to **same space!**

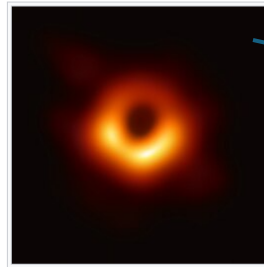
For other uses, see [Black hole \(disambiguation\)](#).

A **black hole** is a region of [spacetime](#) where [gravity](#) is so strong that nothing, including [light](#) and other [electromagnetic waves](#), has enough energy to escape it.^[2] The theory of [general relativity](#) predicts that a sufficiently compact [mass](#) can deform spacetime to form a black hole.^{[3][4]} The [boundary](#) of no escape is called the [event horizon](#). Although it has a great effect on the fate and circumstances of an object crossing it, it has no locally detectable features according to general relativity.^[5] In many ways, a black hole acts like an ideal [black body](#), as it reflects no light.^{[6][7]} Moreover, [quantum field theory in curved spacetime](#) predicts that event horizons emit [Hawking radiation](#), with the same spectrum as a black body of a [temperature](#) inversely proportional to its mass. This temperature is of the order of billionths of a [kelvin](#) for [stellar black holes](#), making it essentially impossible to observe directly.

Objects whose [gravitational fields](#) are too strong for light to escape were first considered in the 18th century by [John Michell](#) and [Pierre-Simon Laplace](#).^[8] In 1916, [Karl Schwarzschild](#) found the first modern solution of general relativity that would characterize a black hole. [David Finkelstein](#), in 1958, first published the interpretation of "black hole" as a region of space from which nothing can escape. Black holes were long considered a mathematical curiosity; it was not until the 1960s that theoretical work showed they were a generic prediction of general relativity. The discovery of [neutron stars](#) by [Jocelyn Bell Burnell](#) in 1967 sparked interest in [gravitationally collapsed](#) compact objects as a possible astrophysical reality. The first black hole known was [Cygnus X-1](#), identified by several researchers independently in 1971.^{[9][10]}

Black holes of stellar mass form when massive stars collapse at the end of their life cycle. After a black hole has formed, it can grow by absorbing mass from its surroundings. [Supermassive black holes](#) of millions of [solar masses](#) (M_{\odot}) may form by absorbing other stars and merging with other black holes. There is consensus that supermassive black holes exist in the centres of most [galaxies](#).

The presence of a black hole can be inferred through its interaction with other [matter](#) and with electromagnetic radiation such as visible light. Any matter that falls onto a black hole can form an external [accretion disk](#) heated by [friction](#), forming [quasars](#), some of the brightest objects in the universe. Stars passing too close to a supermassive black hole can be shredded into streamers that shine very brightly before being "swallowed."^[11] If other stars are orbiting a black hole, their orbits can be used to determine the black hole's mass and location. Such observations can be used to exclude possible alternatives such as neutron stars. In this way, astronomers have identified numerous stellar black hole candidates in [binary systems](#) and established that the radio source known as [Sagittarius A*](#), at the core of the [Milky Way](#) galaxy, contains a supermassive black hole of about 4.3 million [solar masses](#).



Direct radio image of a supermassive black hole at the core of Messier 87^[1]



Animated simulation of a Schwarzschild black hole with a galaxy passing behind. Around the time of alignment, extreme gravitational lensing of the galaxy is observed.

Image Embedding

Text Embedding

Should be close!

Image Embedding

Text Embedding

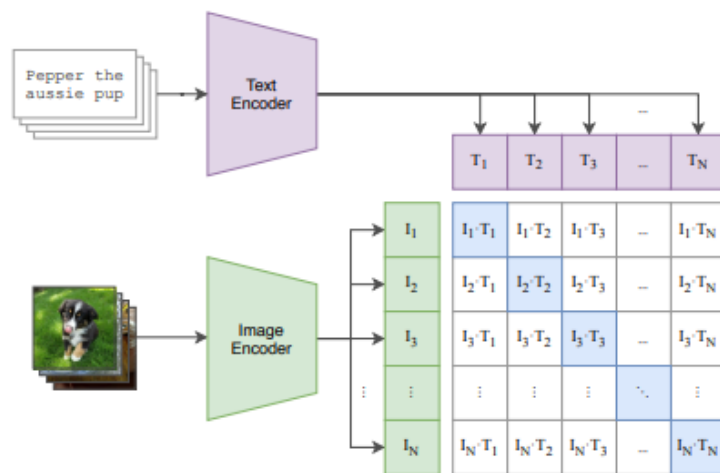
Should be close!

Intuition: Concepts in images and the text should be the same!

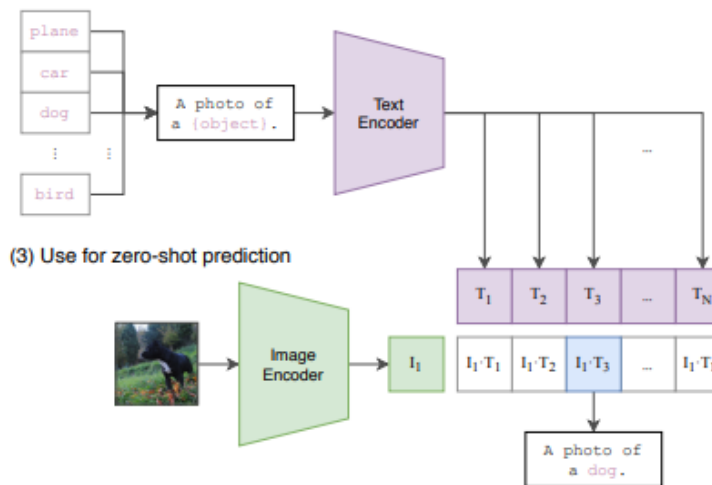
CLIP Paper

(Learning Transferable Visual Models From Natural Language Supervision)

(1) Contrastive pre-training



(2) Create dataset classifier from label text



(3) Use for zero-shot prediction

Figure 1. Summary of our approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes.

Checks All Pretraining Hits

- 1) **Unsupervised**: No need to annotate the images!
- 2) **Unlimited Supply of Examples**: crawl the web!
- 3) **Rich** enough representation!
- 4) DNNs/Transformers

CLIP achieves SoTA with Zero-Shot!

Learning Transferable Visual Models From Natural Language Supervision

Alec Radford^{*1} Jong Wook Kim^{*1} Chris Hallacy¹ Aditya Ramesh¹ Gabriel Goh¹ Sandhini Agarwal¹
Girish Sastry¹ Amanda Askell¹ Pamela Mishkin¹ Jack Clark¹ Gretchen Krueger¹ Ilya Sutskever¹

Abstract

State-of-the-art computer vision systems are trained to predict a fixed set of predetermined object categories. This restricted form of supervision limits their generality and usability since additional labeled data is needed to specify any other visual concept. Learning directly from raw text about images is a promising alternative which leverages a much broader source of supervision. We demonstrate that the simple pre-training task of predicting which caption goes with which image is an efficient and scalable way to learn SOTA image representations from scratch on a dataset of 400 million (image, text) pairs collected from the internet. After pre-training, natural language is used to reference learned visual concepts (or describe new ones) enabling zero-shot transfer of the model to downstream tasks. We study the performance of this approach by benchmarking on over 30 different existing computer vision datasets, spanning tasks such as OCR, action recognition in videos, geo-localization, and many types of fine-grained object classification. The model transfers non-trivially to most tasks and is often competitive with a fully supervised baseline without the need for any dataset specific training. For instance, we match the accuracy of the original ResNet-50 on ImageNet zero-shot without needing to use any of the 1.28 million training examples it was trained on. We release our code and pre-trained model weights at <https://github.com/OpenAI/CLIP>.

1. Introduction and Motivating Work

Pre-training methods which learn directly from raw text have revolutionized NLP over the last few years (Dai & Le, 2015; Peters et al., 2018; Howard & Ruder, 2018; Radford et al., 2018; Devlin et al., 2018; Raffel et al., 2019).

^{*}Equal contribution ¹OpenAI, San Francisco, CA 94110, USA. Correspondence to: <{alec, jongwook}@openai.com>.

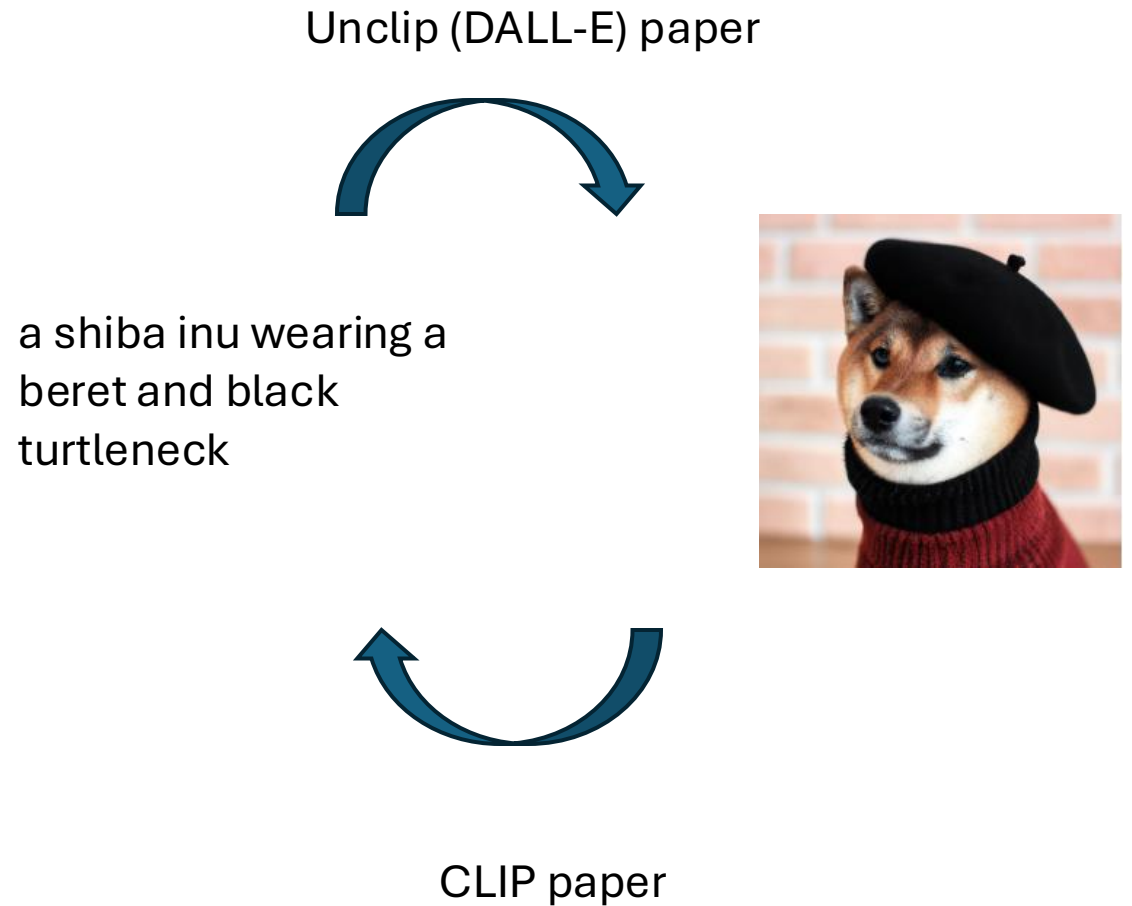
Task-agnostic objectives such as autoregressive and masked language modeling have scaled across many orders of magnitude in compute, model capacity, and data, steadily improving capabilities. The development of “text-to-text” as a standardized input-output interface (McCann et al., 2018; Radford et al., 2019; Raffel et al., 2019) has enabled task-agnostic architectures to zero-shot transfer to downstream datasets removing the need for specialized output heads or dataset specific customization. Flagship systems like GPT-3 (Brown et al., 2020) are now competitive across many tasks with bespoke models while requiring little to no dataset specific training data.

These results suggest that the aggregate supervision accessible to modern pre-training methods within web-scale collections of text surpasses that of high-quality crowd-labeled NLP datasets. However, in other fields such as computer vision it is still standard practice to pre-train models on crowd-labeled datasets such as ImageNet (Deng et al., 2009). Could scalable pre-training methods which learn directly from web text result in a similar breakthrough in computer vision? Prior work is encouraging.

Over 20 years ago Mori et al. (1999) explored improving content based image retrieval by training a model to predict the nouns and adjectives in text documents paired with images. Quattoni et al. (2007) demonstrated it was possible to learn more data efficient image representations via manifold learning in the weight space of classifiers trained to predict words in captions associated with images. Srivastava & Salakhutdinov (2012) explored deep representation learning by training multimodal Deep Boltzmann Machines on top of low-level image and text tag features. Joulin et al. (2016) modernized this line of work and demonstrated that CNNs trained to predict words in image captions learn useful image representations. They converted the title, description, and hashtag metadata of images in the YFCC100M dataset (Thomee et al., 2016) into a bag-of-words multi-label classification task and showed that pre-training AlexNet (Krizhevsky et al., 2012) to predict these labels learned representations which preformed similarly to ImageNet-based pre-training on transfer tasks. Li et al. (2017) then extended this approach to predicting phrase n-grams in addition to individual words and demonstrated the ability of their system to zero-shot transfer to other image

- Gets nearly 80% accuracy on Imagenet with zero-shot!!!
- That is no training on Imagenet!

Similar concepts in
different modalities
are represented by
similar vectors



Diffusion models, next word prediction, DINO all have the same principle: unsupervised learning

Take a piece of data → Corrupt it in a controlled way → Reconstruct it via a transformer

Difference between GPT3 and GPT4?

Prompt *Explain the moon landing to a 6 year old in a few sentences.*

Completion GPT-3

Explain the theory of gravity to a 6 year old.

<

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.

InstructGPT

People went to the moon, and they took pictures of what they saw, and sent them back to the earth so we could all see them.

Difference between GPT3 and GPT4?

Training language models to follow instructions with human feedback

Long Ouyang* Jeff Wu* Xu Jiang* Diogo Almeida* Carroll L. Wainwright*
Pamela Mishkin* Chong Zhang Sandhini Agarwal Katarina Slama Alex Ray
John Schulman Jacob Hilton Fraser Kelton Luke Miller Maddie Simens
Amanda Askell† Peter Welinder Paul Christiano*‡
Jan Leike* Ryan Lowe*

OpenAI

Abstract

Making language models bigger does not inherently make them better at following a user's intent. For example, large language models can generate outputs that are untruthful, toxic, or simply not helpful to the user. In other words, these models are not *aligned* with their users. In this paper, we show an avenue for aligning language models with user intent on a wide range of tasks by fine-tuning with human feedback. Starting with a set of labeler-written prompts and prompts submitted through the OpenAI API, we collect a dataset of labeler demonstrations of the desired model behavior, which we use to fine-tune GPT-3 using supervised learning. We then collect a dataset of rankings of model outputs, which we use to further fine-tune this supervised model using reinforcement learning from human feedback. We call the resulting models *InstructGPT*. In human evaluations on our prompt distribution, outputs from the 1.3B parameter InstructGPT model are preferred to outputs from the 175B GPT-3, despite having 100x fewer parameters. Moreover, InstructGPT models show improvements in truthfulness and reductions in toxic output generation while having minimal performance regressions on public NLP datasets. Even though InstructGPT still makes simple mistakes, our results show that fine-tuning with human feedback is a promising direction for aligning language models with human intent.

GPT4 does another round of training specifically teaching how to follow instructions or interact with the users.

This phase is called *Alignment*, RLHF finetuning, etc.

The model's *knowledge* is fixed during the pretraining, but this phase teaches human preferences.

Open Problems

- Diffusion process is very successful for images and nextword prediction for language.
Is there a unifying process?
- What other new unsupervised learning tasks we can create?
- LLMs are not good in reasoning, planning, etc. What new unsupervised learning processes that can unlock these?
- Math: We do next word prediction for math; are there better processes?
(to me, math is a different modality compared to text)
- Ilya Sutskever “An Observation on Generalization”
 - any abstraction is a form of compression; can we formalize it?

https://www.youtube.com/watch?v=AKMuA_TVz3A&list=PLgKuh-lKre12qVTl88k2n2N37tT-BpmHT&index=4

Three Breakthrough Ideas

Unsupervised or self-supervised
Learning

Transformer Architecture

Scale

Scaling -> improves next word prediction accuracy -> better in-context learning

(one of the time tested principles)

From GPT3 paper

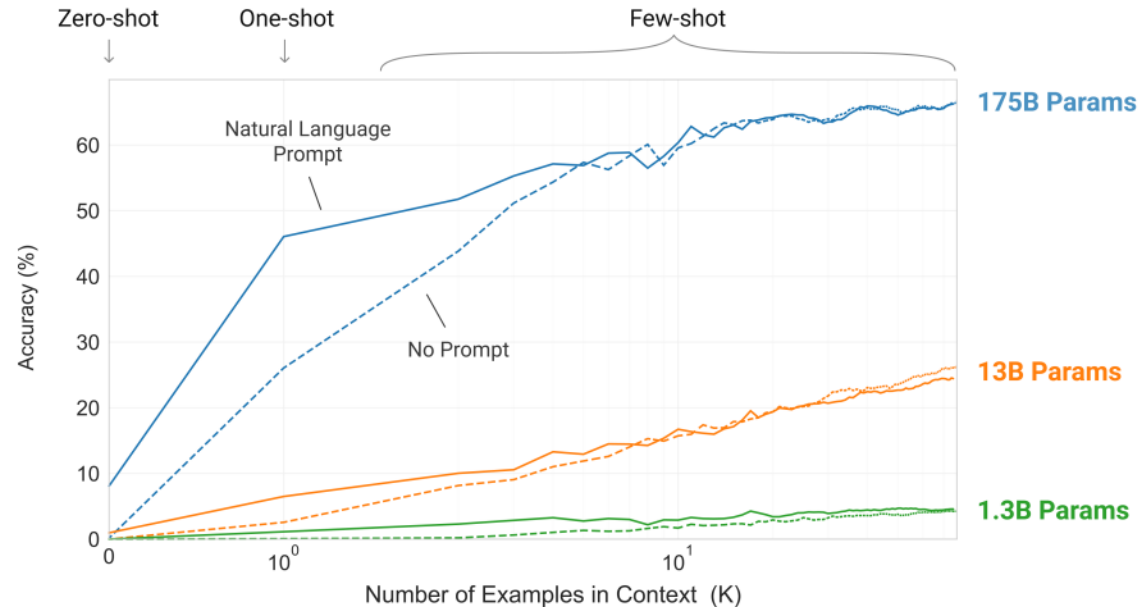
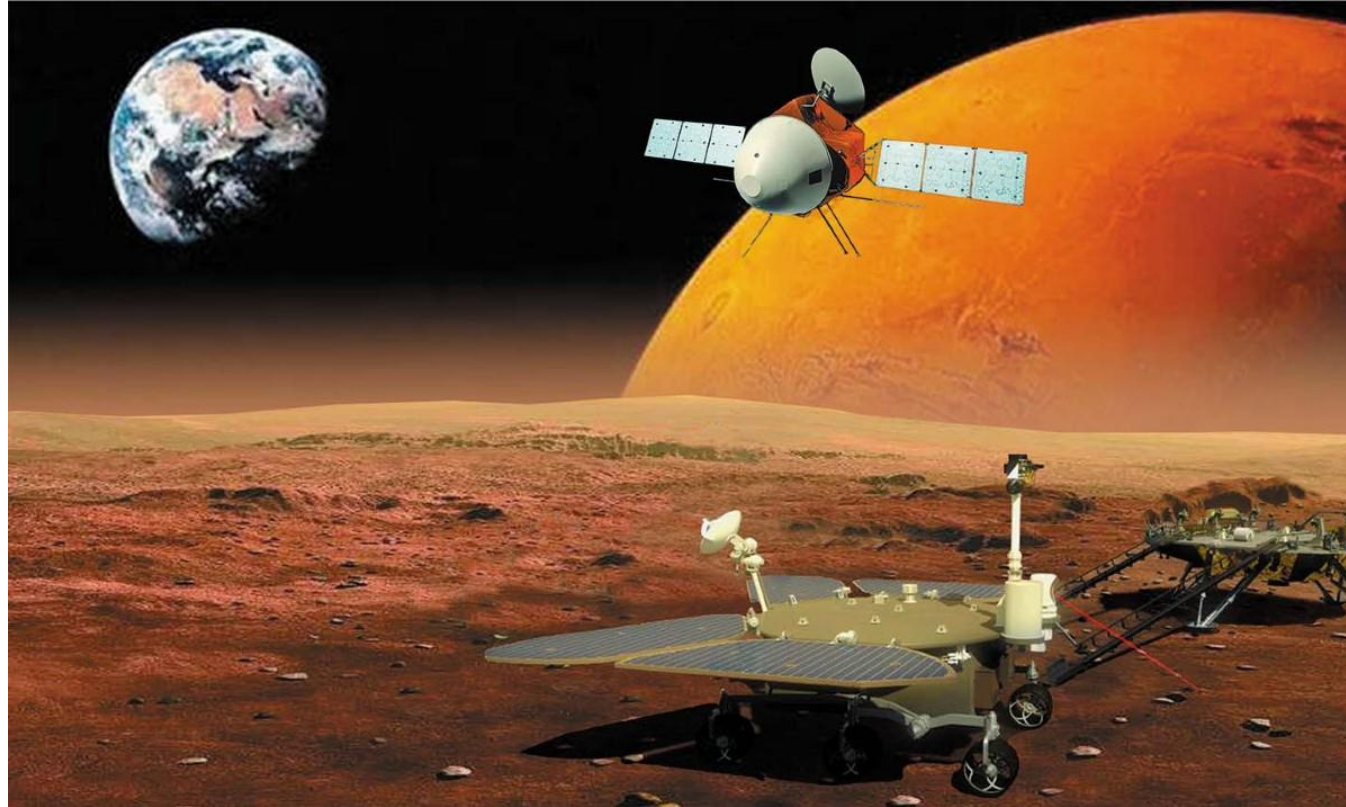


Figure 1.2: Larger models make increasingly efficient use of in-context information. We show in-context learning performance on a simple task requiring the model to remove random symbols from a word, both with and without a natural language task description (see Sec. 3.9.2). The steeper “in-context learning curves” for large models demonstrate improved ability to learn a task from contextual information. We see qualitatively similar behavior across a wide range of tasks.

sufficient to enable a human to perform a new task to at least a reasonable degree of competence. Aside from pointing to a conceptual limitation in our current NLP techniques, this adaptability has practical advantages – it allows humans to seamlessly mix together or switch between many tasks and skills, for example performing addition during a lengthy dialogue. To be broadly useful, we would someday like our NLP systems to have this same fluidity and generality.

Training neural networks is (very) expensive



So, how should we select how large is the model, how much data I should use, and how long I should train?

Scaling Laws for Neural Language Models

Jared Kaplan *

Johns Hopkins University, OpenAI

jaredk@jhu.edu

Sam McCandlish*

OpenAI

sam@openai.com

Tom Henighan

OpenAI

henighan@openai.com

Tom B. Brown

OpenAI

tom@openai.com

Benjamin Chess

OpenAI

bchess@openai.com

Rewon Child

OpenAI

rewon@openai.com

Scott Gray

OpenAI

scott@openai.com

Alec Radford

OpenAI

alec@openai.com

Jeffrey Wu

OpenAI

jeffwu@openai.com

Dario Amodei

OpenAI

damodei@openai.com

Abstract

We study empirical scaling laws for language model performance on the cross-entropy loss. The loss scales as a power-law with model size, dataset size, and the amount of compute used for training, with some trends spanning more than seven orders of magnitude. Other architectural details such as network width or depth have minimal effects within a wide range. Simple equations govern the dependence of overfitting on model/dataset size and the dependence of training speed on model size. These relationships allow us to determine the optimal allocation of a fixed compute budget. Larger models are significantly more sample-efficient, such that optimally compute-efficient training involves training very large models on a relatively modest amount of data and stopping significantly before convergence.



Training Compute-Optimal Large Language Models

Jordan Hoffmann*, Sebastian Borgeaud*, Arthur Mensch*, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals and Laurent Sifre*

*Equal contributions

We investigate the optimal model size and number of tokens for training a transformer language model under a given compute budget. We find that current large language models are significantly under-trained, a consequence of the recent focus on scaling language models whilst keeping the amount of training data constant. By training over 400 language models ranging from 70 million to over 16 billion parameters on 5 to 500 billion tokens, we find that for compute-optimal training, the model size and the number of training tokens should be scaled equally: for every doubling of model size the number of training tokens should also be doubled. We test this hypothesis by training a predicted compute-optimal model, *Chinchilla*, that uses the same compute budget as *Gopher* but with 70B parameters and 4× more data. *Chinchilla* uniformly and significantly outperforms *Gopher* (280B), GPT-3 (175B), Jurassic-1 (178B), and Megatron-Turing NLG (530B) on a large range of downstream evaluation tasks. This also means that *Chinchilla* uses substantially less compute for fine-tuning and inference, greatly facilitating downstream usage. As a highlight, *Chinchilla* reaches a state-of-the-art average accuracy of 67.5% on the MMLU benchmark, greater than a 7% improvement over *Gopher*.

Hypothesis (verified time and again by several models)

Pretraining loss for next word prediction is a good indicator of model performance in the wild.

Scaling Laws Problem

In this work, we revisit the question: *Given a fixed FLOPs budget,¹ how should one trade-off model size and the number of training tokens?* To answer this question, we model the final pre-training loss² $L(N, D)$ as a function of the number of model parameters N , and the number of training tokens, D . Since the computational budget C is a deterministic function $\text{FLOPs}(N, D)$ of the number of seen training tokens and model parameters, we are interested in minimizing L under the constraint $\text{FLOPs}(N, D) = C$:

$$N_{\text{opt}}(C), D_{\text{opt}}(C) = \underset{N, D \text{ s.t. } \text{FLOPs}(N, D) = C}{\text{argmin}} L(N, D). \quad (1)$$

The functions $N_{\text{opt}}(C)$, and $D_{\text{opt}}(C)$ describe the optimal allocation of a computational budget C . We

Kaplan et al

Smooth power laws: Performance has a power-law relationship with each of the three scale factors N, D, C when not bottlenecked by the other two, with trends spanning more than six orders of magnitude (see Figure 1). We observe no signs of deviation from these trends on the upper end, though performance must flatten out eventually before reaching zero loss. (Section 3)

Chinchilla paper

Efficient frontier. We can approximate the functions N_{opt} and D_{opt} by minimizing the parametric loss \hat{L} under the constraint $\text{FLOPs}(N, D) \approx 6ND$ (Kaplan et al., 2020). The resulting N_{opt} and D_{opt} balance the two terms in Equation (3) that depend on model size and data. By construction, they have a power-law form:

$$N_{opt}(C) = G \left(\frac{C}{6} \right)^a, \quad D_{opt}(C) = G^{-1} \left(\frac{C}{6} \right)^b, \quad \text{where} \quad G = \left(\frac{\alpha A}{\beta B} \right)^{\frac{1}{\alpha+\beta}}, \quad a = \frac{\beta}{\alpha+\beta}, \quad \text{and} \quad b = \frac{\alpha}{\alpha+\beta}. \quad (4)$$

We show contours of the fitted function \hat{L} in Figure 4 (left), and the closed-form efficient computational frontier in blue. From this approach, we find that $a = 0.46$ and $b = 0.54$ —as summarized in Table 2.

Table 2 | **Estimated parameter and data scaling with increased training compute.** The listed values are the exponents, a and b , on the relationship $N_{opt} \propto C^a$ and $D_{opt} \propto C^b$. Our analysis suggests a near equal scaling in parameters and data with increasing compute which is in clear contrast to previous work on the scaling of large models. The 10th and 90th percentiles are estimated via bootstrapping data (80% of the dataset is sampled 100 times) and are shown in parenthesis.

Approach	Coeff. a where $N_{opt} \propto C^a$	Coeff. b where $D_{opt} \propto C^b$
1. Minimum over training curves	0.50 (0.488, 0.502)	0.50 (0.501, 0.512)
2. IsoFLOP profiles	0.49 (0.462, 0.534)	0.51 (0.483, 0.529)
3. Parametric modelling of the loss	0.46 (0.454, 0.455)	0.54 (0.542, 0.543)
Kaplan et al. (2020)	0.73	0.27

Open Problems

Theoretical understanding of scaling laws and their implications.

A Theory for Emergence of Complex Skills in Language Models
[Sanjeev Arora](#), [Anirudh Goyal](#)

Tensor Programs V: Tuning Large Neural Networks via Zero-Shot Hyperparameter Transfer

Greg Yang^{*×} Edward J. Hu^{*×†} Igor Babuschkin[°] Szymon Sidor[°] Xiaodong Liu[×]
David Farhi[°] Nick Ryder[°] Jakub Pachocki[°] Weizhu Chen[×] Jianfeng Gao[×]
[×]Microsoft Corporation [°]OpenAI

Abstract

Hyperparameter (HP) tuning in deep learning is an expensive process, prohibitively so for neural networks (NNs) with billions of parameters. We show that, in the recently discovered Maximal Update Parametrization (μP), many optimal HPs remain stable even as model size changes. This leads to a new HP tuning paradigm we call $\mu Transfer$: parametrize the target model in μP , tune the HP indirectly on a smaller model, and *zero-shot transfer* them to the full-sized model, i.e., without directly tuning the latter at all. We verify $\mu Transfer$ on Transformer and ResNet. For example, 1) by transferring pretraining HPs from a model of 13M parameters, we outperform published numbers of BERT-large (350M parameters), with a total tuning cost equivalent to pretraining BERT-large once; 2) by transferring from 40M parameters, we outperform published numbers of the 6.7B GPT-3 model, with tuning cost only 7% of total pretraining cost. A Pytorch implementation of our technique can be found at github.com/microsoft/mup and installable via `pip install mup`.

1 Introduction

Hyperparameter (HP) tuning is critical to deep learning. Poorly chosen HPs result in subpar performance and training instability. Many published baselines are hard to compare to one another due to varying degrees of HP tuning. These issues are exacerbated when training extremely large deep learning models, since state-of-the-art networks with billions of parameters become prohibitively expensive to tune.

Recently, [57] showed that different neural network parametrizations induce different infinite-width limits and proposed the *Maximal Update Parametrization* (abbreviated μP) (summarized in Table 3) that enables “maximal” feature learning in the limit. Intuitively, it ensures that each layer is updated on the same order during training *regardless of width*.² In contrast, while the standard parametrization (SP) ensures activations are of unit order at initialization, it actually causes them to blow up in wide models during training [57] essentially due to an imbalance of per-layer

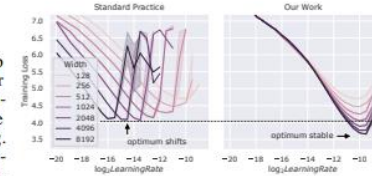


Figure 1: Training loss against learning rate on Transformers of varying d_{model} trained with Adam. Conventionally and in contrast with our technique, different widths do not share the same optimal hyperparameter; wider networks do not always perform better than narrower ones; in fact they underperform the same-width networks in our technique even after tuning learning rate (see dashed line). See Sections 3 and 4 for experimental setup.

¹Work done partly during Microsoft AI Residency Program.

^{*}Equal contribution. Order is random. Correspondence to {gregyang, edwardhu}@microsoft.com

²i.e., the updates’ effect on activations becomes roughly independent of width in the large width limit.

Other forms of scaling laws:

Hyperparameters scaling, initializations, etc,

Physics of LLMs: Allen Zhu and Li.

Information storage capacity of transformers:

2-bits per parameter

(<https://arxiv.org/abs/2404.05405>)

Three Breakthrough Ideas

Unsupervised or self-supervised
Learning

Transformer Architecture

Scale

Attention is All You Need

<https://arxiv.org/abs/1706.03762>

Transformer

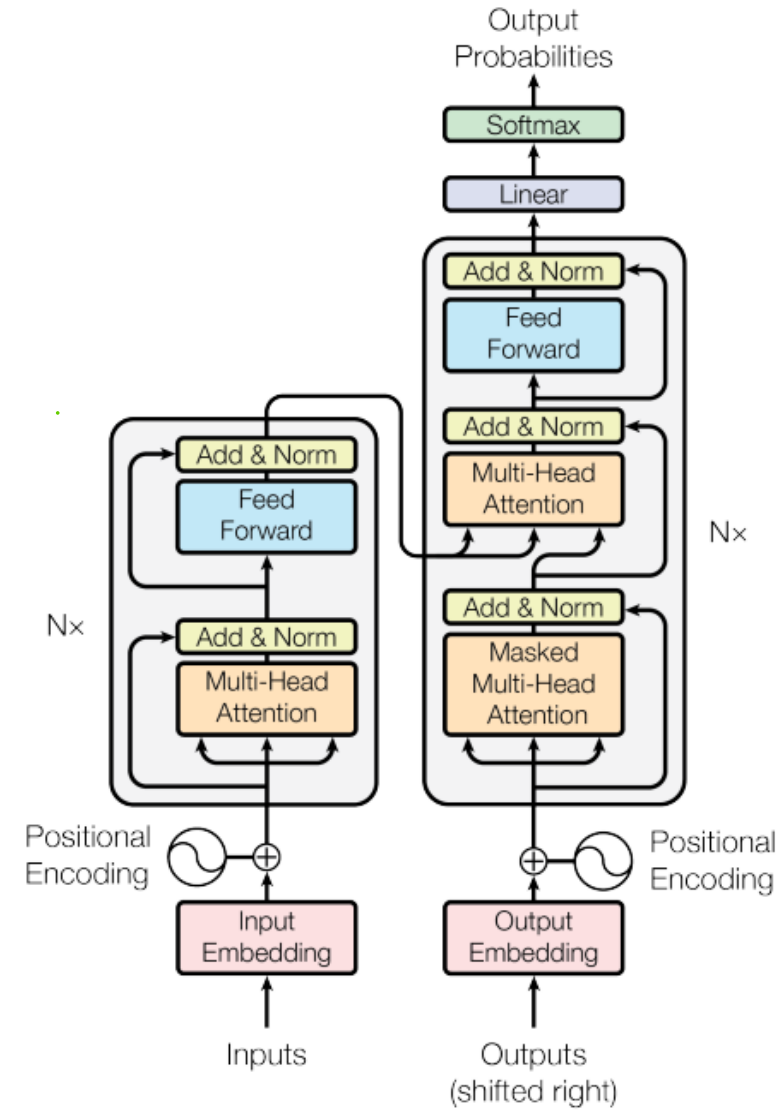


Figure 1: The Transformer - model architecture.

Attention is All You Need

<https://arxiv.org/abs/1706.03762>

Transformer

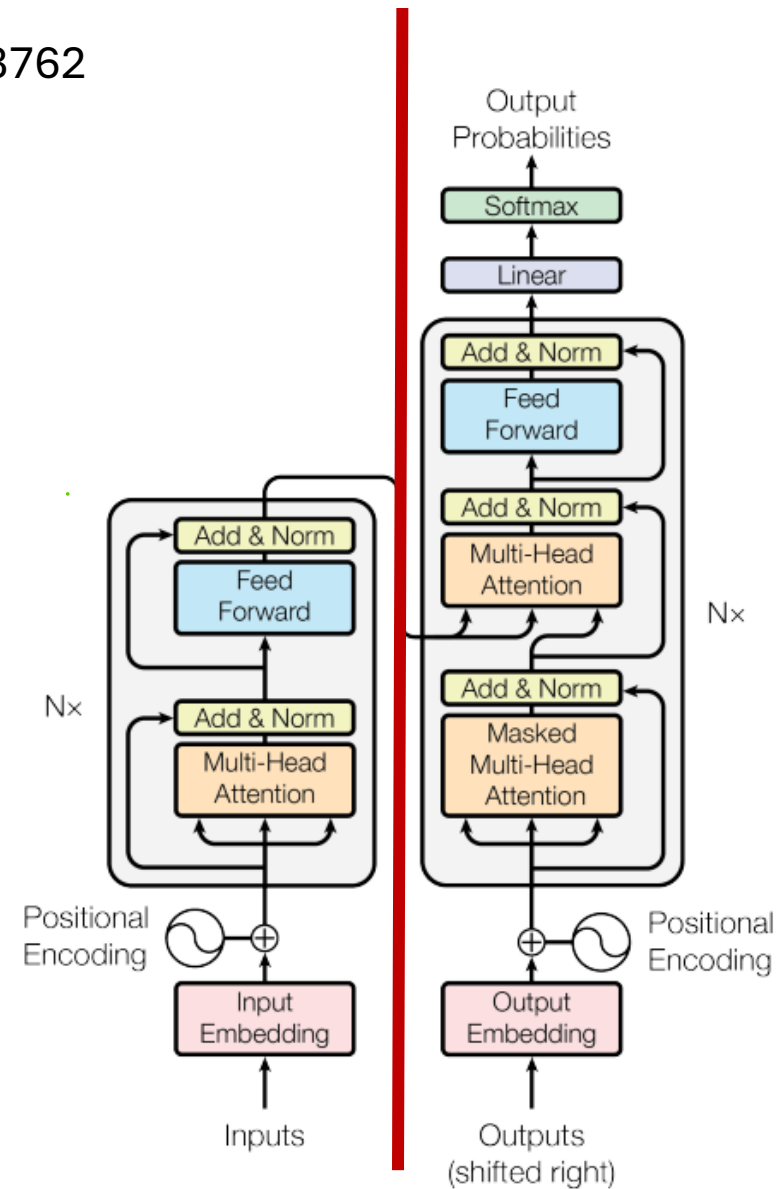
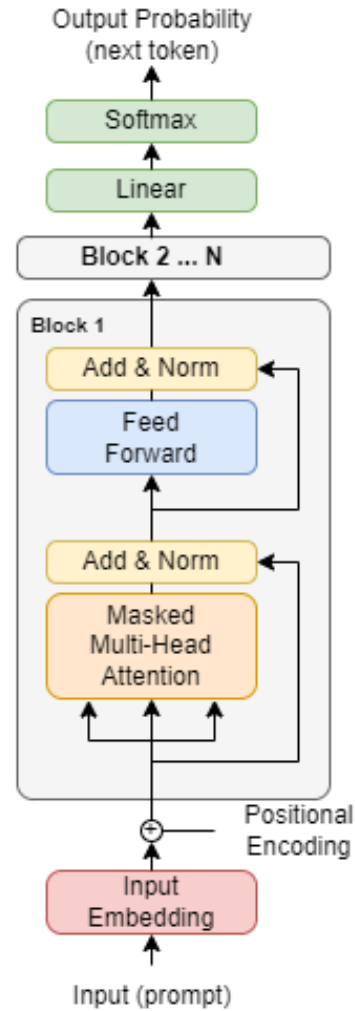
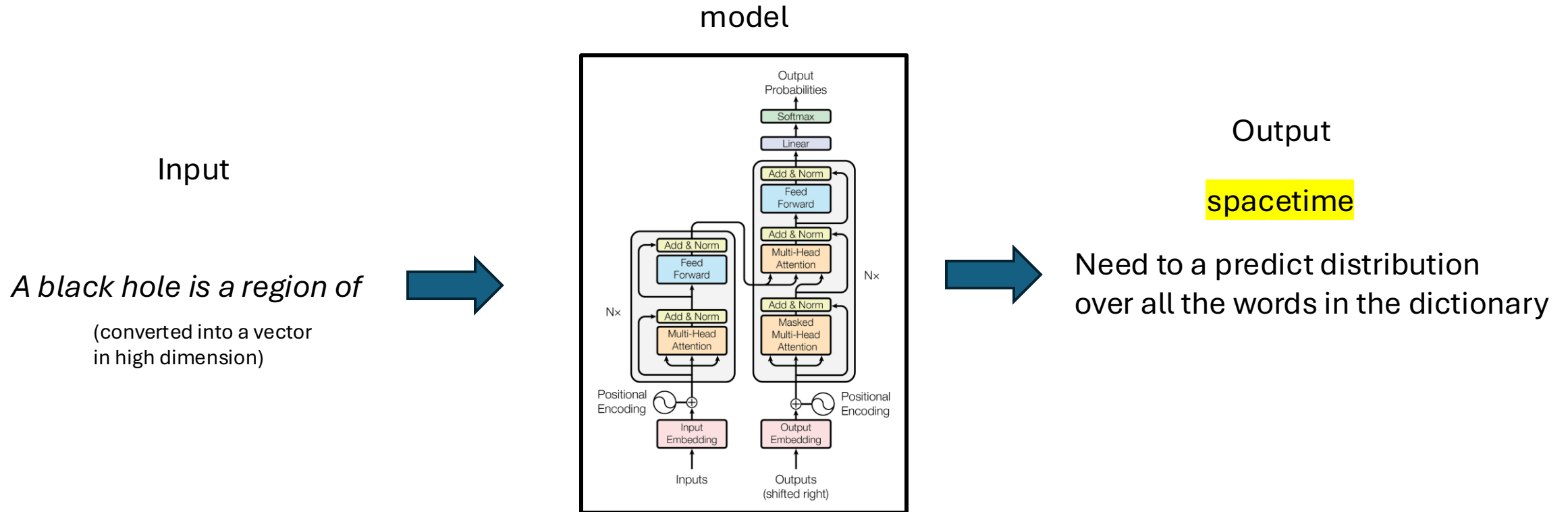


Figure 1: The Transformer - model architecture.

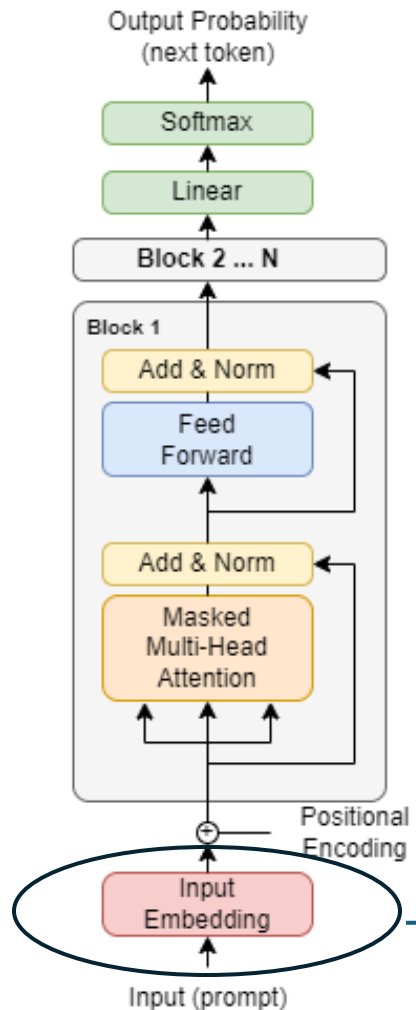
Decoder only Transformer (All GPTs, LLaMAs, Geminis, all good models)



What is Input and Output?



Embedding Layers: Convert sequence of words into a matrix, where each word (or token) is represented as a vector



- Input Embedding Layer: Just a linear layer (matrix multiplication) to learn better representation of words/tokens.
- The embeddings of tokens should better represent relationships among tokens or words.
- Ex: queen and king should have closer representation
Similarly, (queen – king) should nearly same (woman – man)

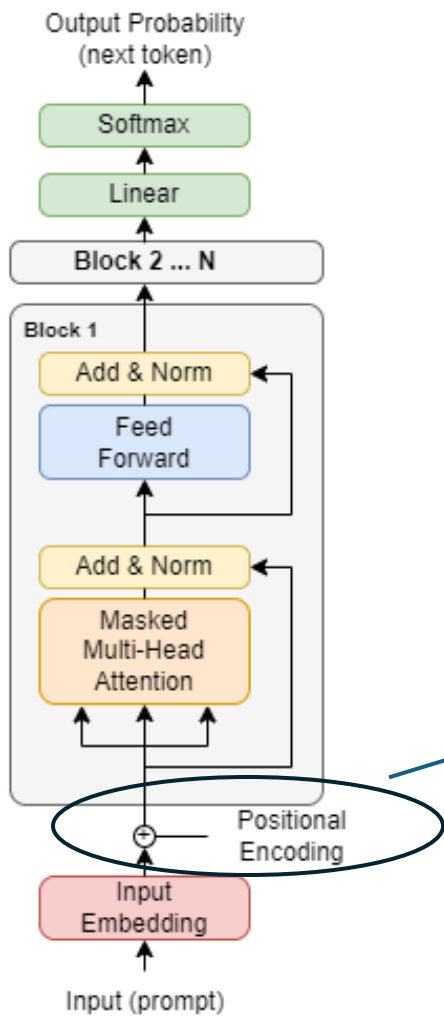
Input sequence of tokens = input matrix



Input matrix * $W_{\text{embedding}}$ matrix

Each token in the input sequence has a new representation in a smaller dimension

Positional Embedding



Input sequence of tokens = input matrix



Input matrix * $W_{\text{embedding}}$ matrix = idx



For every token, we add a positional encoding of that token. Intuition is that in any sentence, position of the words matter, so that information needs to be encoded in the input.

Example of positional encoding

1. **Sinusoidal Positional Encodings:** Instead of learning separate embeddings for each position in the sequence, sinusoidal positional encodings use sine and cosine functions with different frequencies to encode positional information.

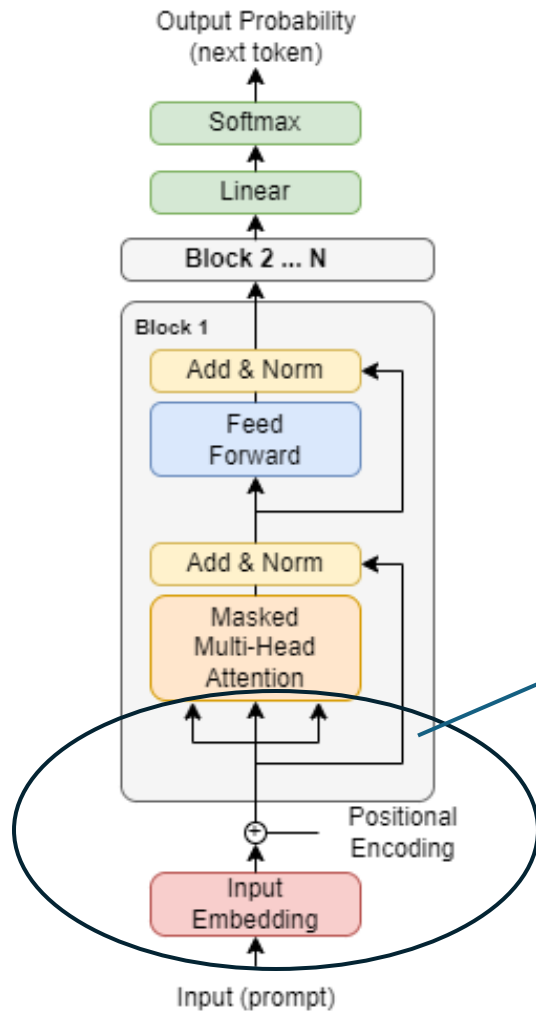
2. **Formulation:** The positional embedding for position i and dimension j is calculated as follows:

$$PE(i, j) = \begin{cases} \sin\left(\frac{i}{10000^{2j/d_{\text{model}}}}\right) & \text{if } j \text{ is even} \\ \cos\left(\frac{i}{10000^{(2j-1)/d_{\text{model}}}}\right) & \text{if } j \text{ is odd} \end{cases}$$

Where:

- $PE(i, j)$ is the positional embedding for position i and dimension j .
- d_{model} is the dimensionality of the embeddings.

Embedding Layers = a good representation of words + their positions



Input sequence of words / tokens = input matrix

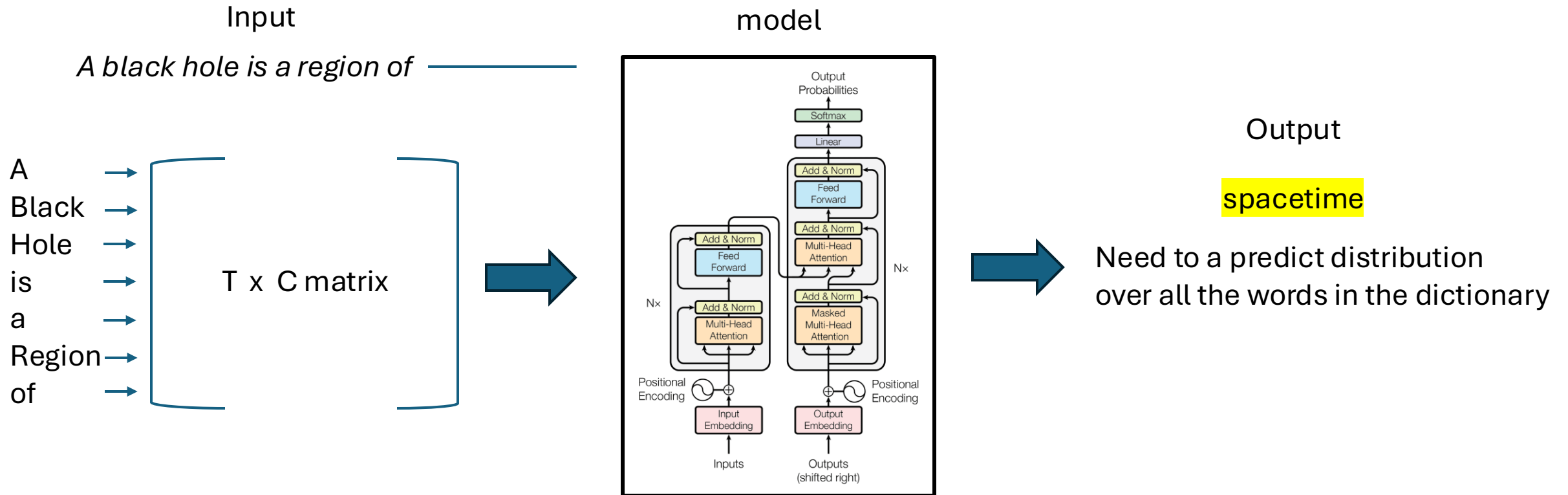
Input matrix * $W_{\text{embedding}}$ matrix = x

x + positional encoding matrix (not learnt)

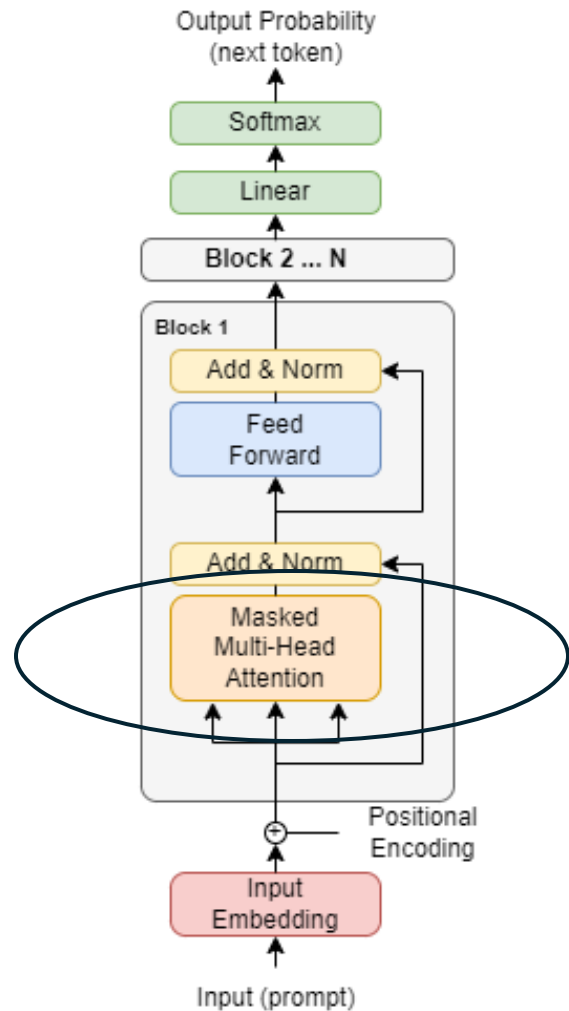
For every token, we add a positional encoding of that token. That, at this point, each token is sum of two vectors: learnt embedding of the token + its position in the sequence. This completes the representation of the input.

X is matrix of size $T * C$, where T is the number of tokens in the sequence and C is the dimension of each vector representing each token.

What is Input and Output?



Attention Function



→ The most important concept or layer in Transformers.
Let us first simplify and understand *single* attention head.
Also, for a moment ignore masked part. We will come to it naturally.


Attention Function

X is matrix of size $T * C$, where T is the number of words in the sequence and C is the dimension of each vector representing each word/token.

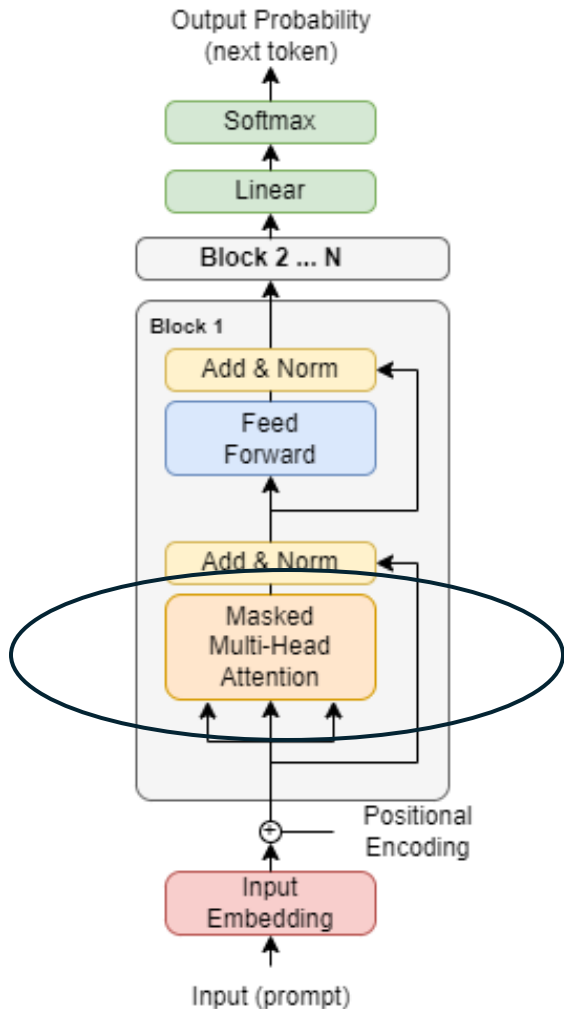
Lets fix a single word and see what attention function does to that word.
Here “of” is our word. *Just remember that every word has a C length vector now*

Our running example sentence:

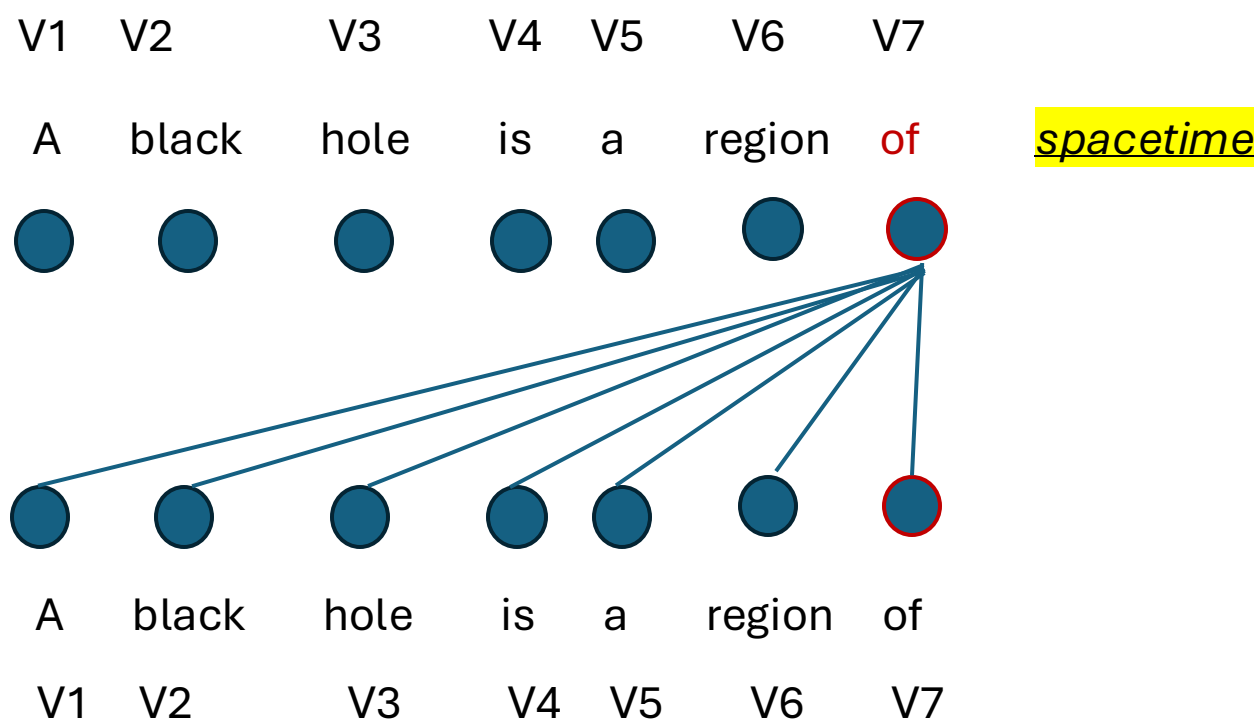
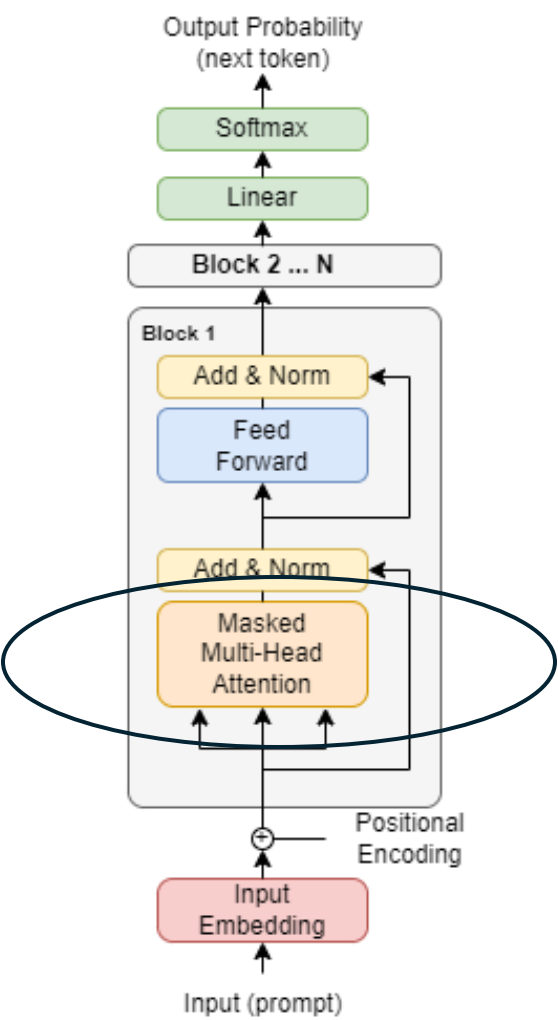
A black hole is a region of spacetime



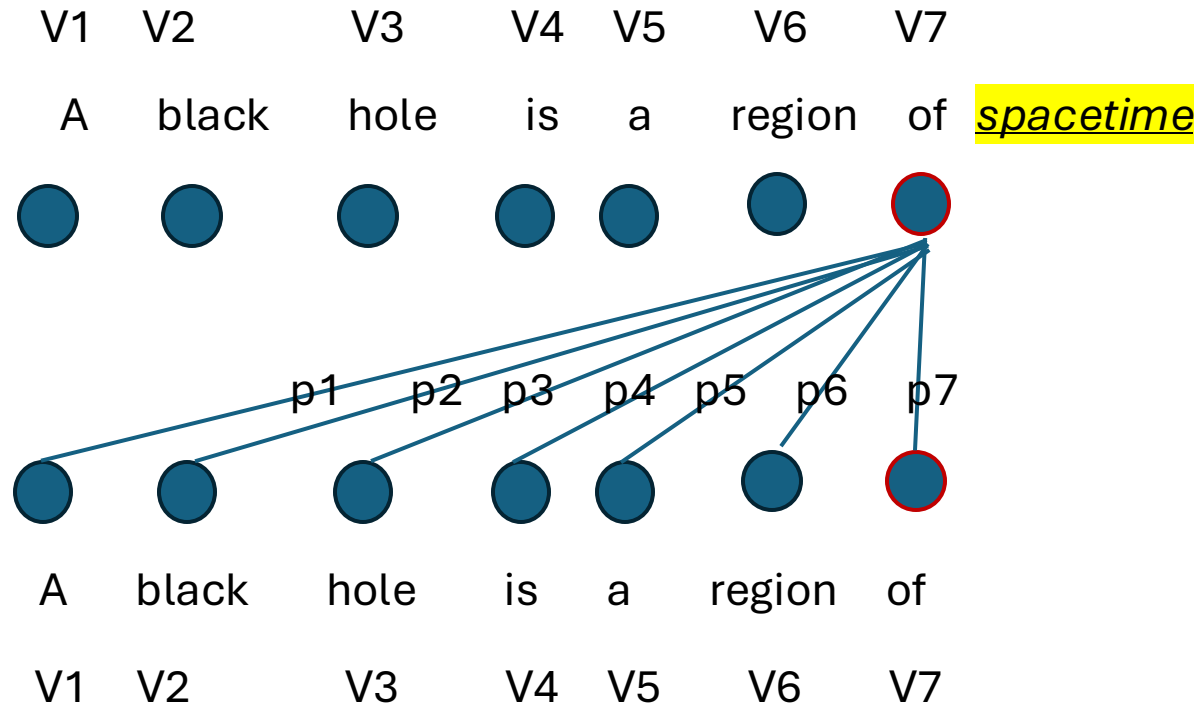
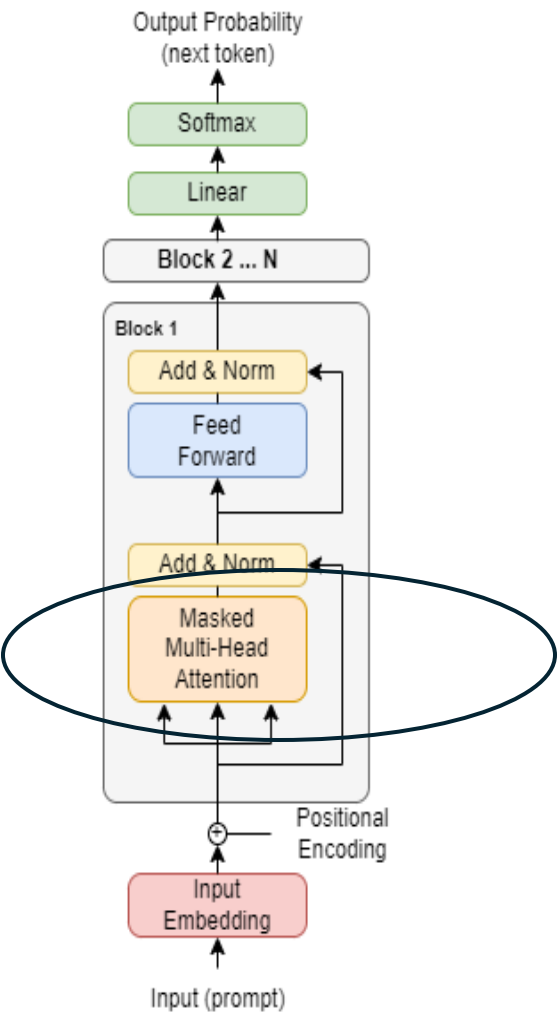
- To predict the word spacetime successfully, we should aggregate all the information until that point that may maximize our chance.
- What are reasonable ways of information aggregation?



Attention Function



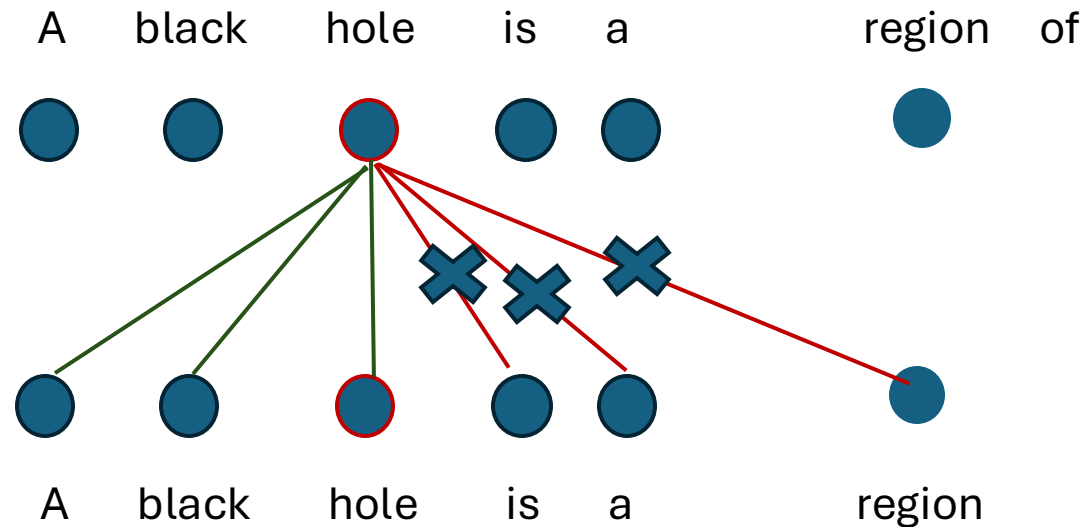
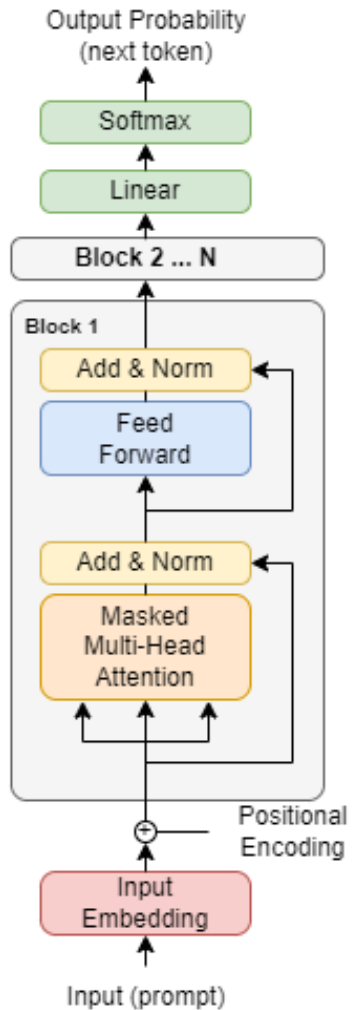
Attention Function



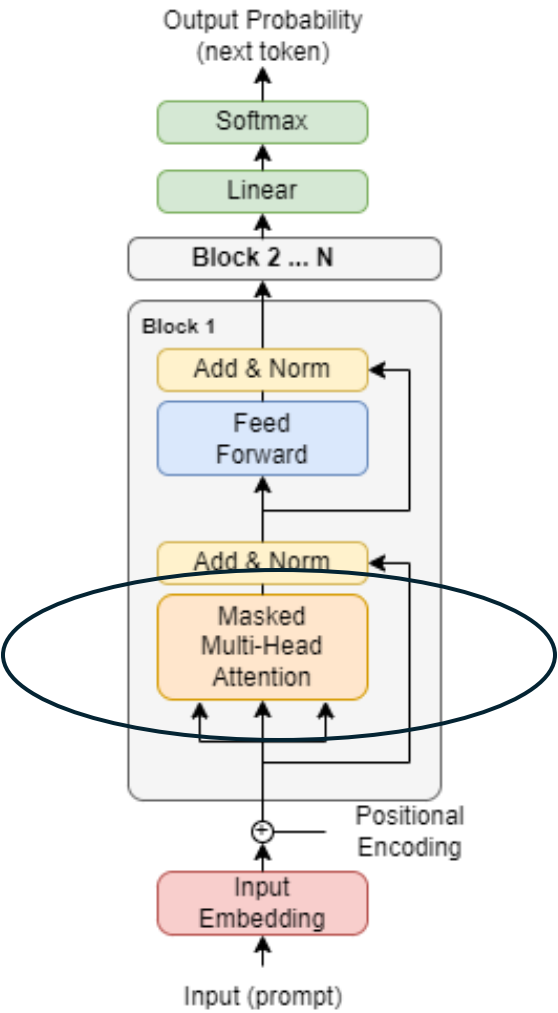
Updated $V7 = \sum_{i} p_i * V_i$

1. Compute dot product with all tokens before it
2. Convert these values to a probability distribution using softmax function
3. Update the representation of the word using convex combination of other tokens

Causal language modeling or autoregressive language modeling = A word can only look at previous tokens

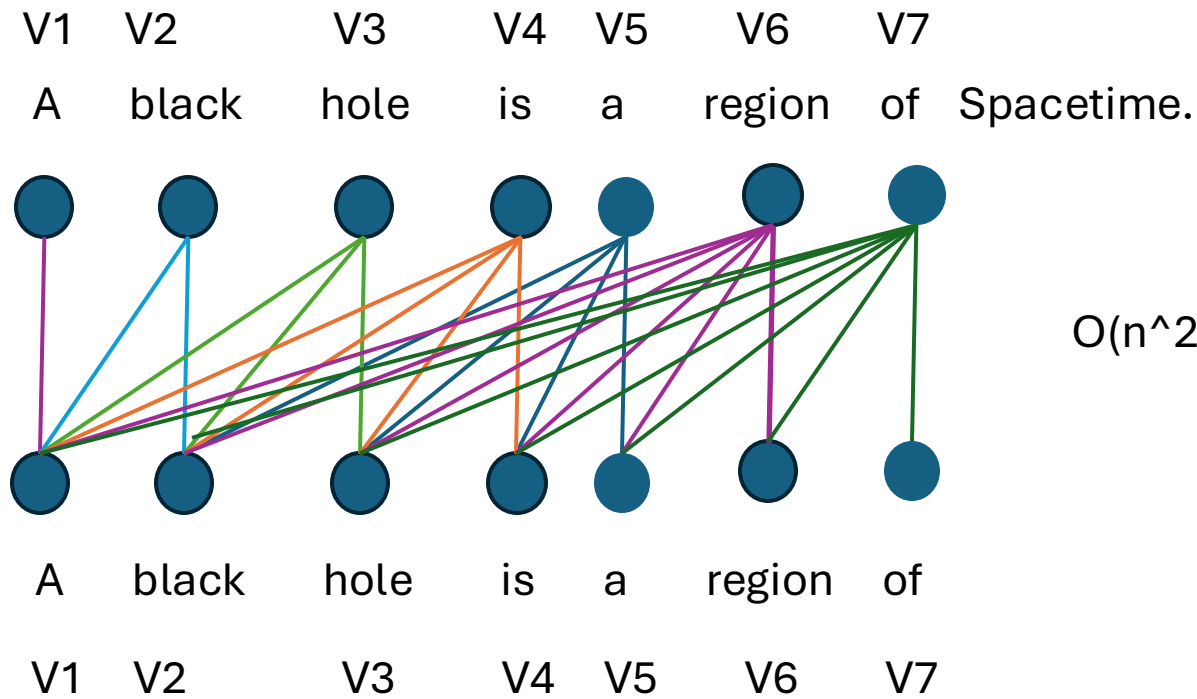


Attention Function



- X is matrix of size $T * C$, where T is the number of tokens in the sequence and C is the dimension of each vector representing each token.
- Every token is trying to predict the next token in the sequence => can only use information from previous tokens (masked language modelling)

SELF ATTENTION



$$\text{Updated } V_j = \sum_{i=1}^{j-1} p_i * V_i$$

Scaled Dot-Product Attention

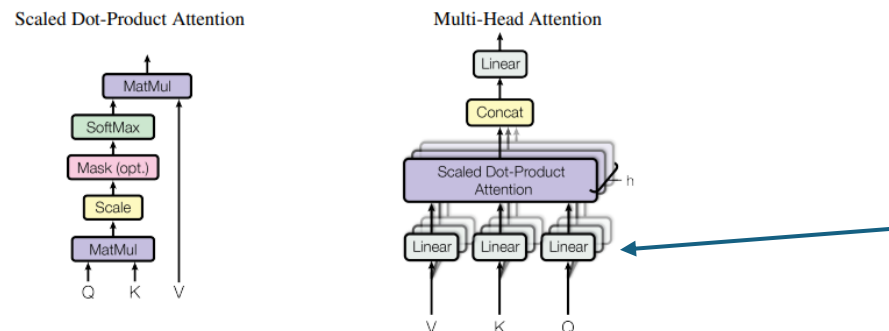


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

3.2.1 Scaled Dot-Product Attention

We call our particular attention "Scaled Dot-Product Attention" (Figure 2). The input consists of queries and keys of dimension d_k , and values of dimension d_v . We compute the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values.

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q . The keys and values are also packed together into matrices K and V . We compute the matrix of outputs as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

The two most commonly used attention functions are additive attention [2], and dot-product (multiplicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$. Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

While for small values of d_k the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of d_k [3]. We suspect that for large values of d_k , the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients⁴. To counteract this effect, we scale the dot products by $\frac{1}{\sqrt{d_k}}$.

Q, K and V are all the same matrices representing input sequence of vectors

After the application of first layer of self-attention, every token representation gets updated: It is a convex combination of softmax applied to dot product of the all previous tokens.

Note that in self-attention every token attends to previous tokens, not to forward ones. It makes sense; we are predicting the next word!!!!

This is all good: But what are we learning?

There are no learning parameters in this attention computation!!!!

Actual Self-Attention

X is matrix of size $T * C$, where T is the number of tokens in the sequence and C is the dimension of each vector representing each token.

1. First, we do a linear transformation of x (multiply by matrices)

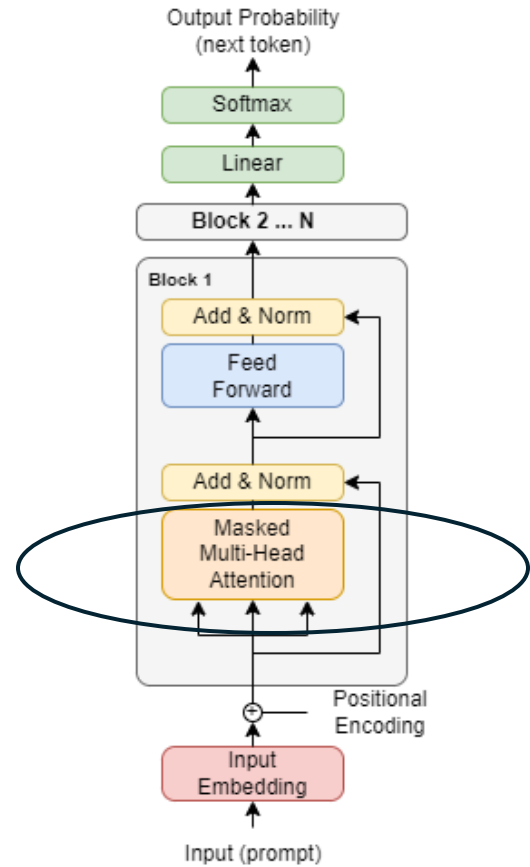
$$\text{Key} = x * W_{\text{key}}$$

$$\text{Query} = x * W_{\text{query}}$$

$$\text{Value} = x * W_{\text{value}}$$

2. Then, we compute the self-attention:

$$\text{Attention}(\text{Query}, \text{Key}, \text{Value}) = \text{softmax}(\text{Query} * \text{Key}^{\text{trans}} / \sqrt{d_k}) * \text{Value}$$



Actual Self-Attention

X is matrix of size $T * C$, where T is the number of tokens in the sequence and C is the dimension of each vector representing each token.

1. First, we do a linear transformation of x (multiply by matrices)

$$\text{Key} = x * W_{\text{key}}$$

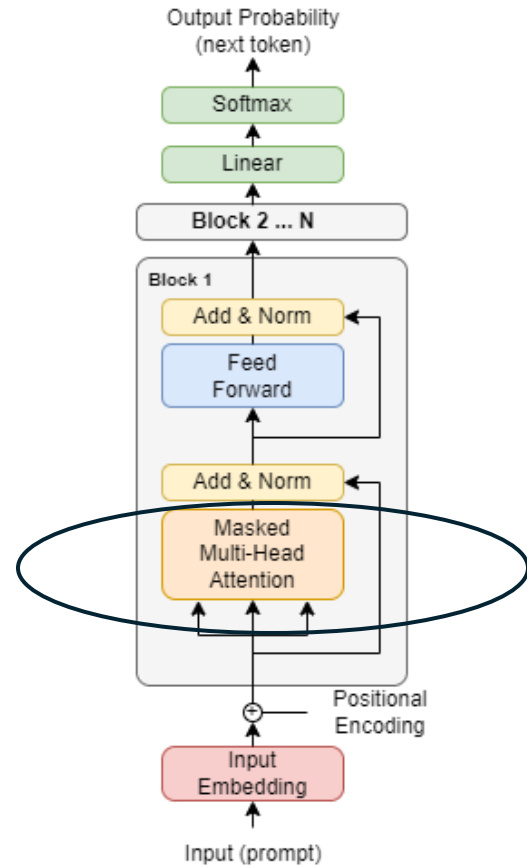
$$\text{Query} = x * W_{\text{query}}$$

$$\text{Value} = x * W_{\text{value}}$$

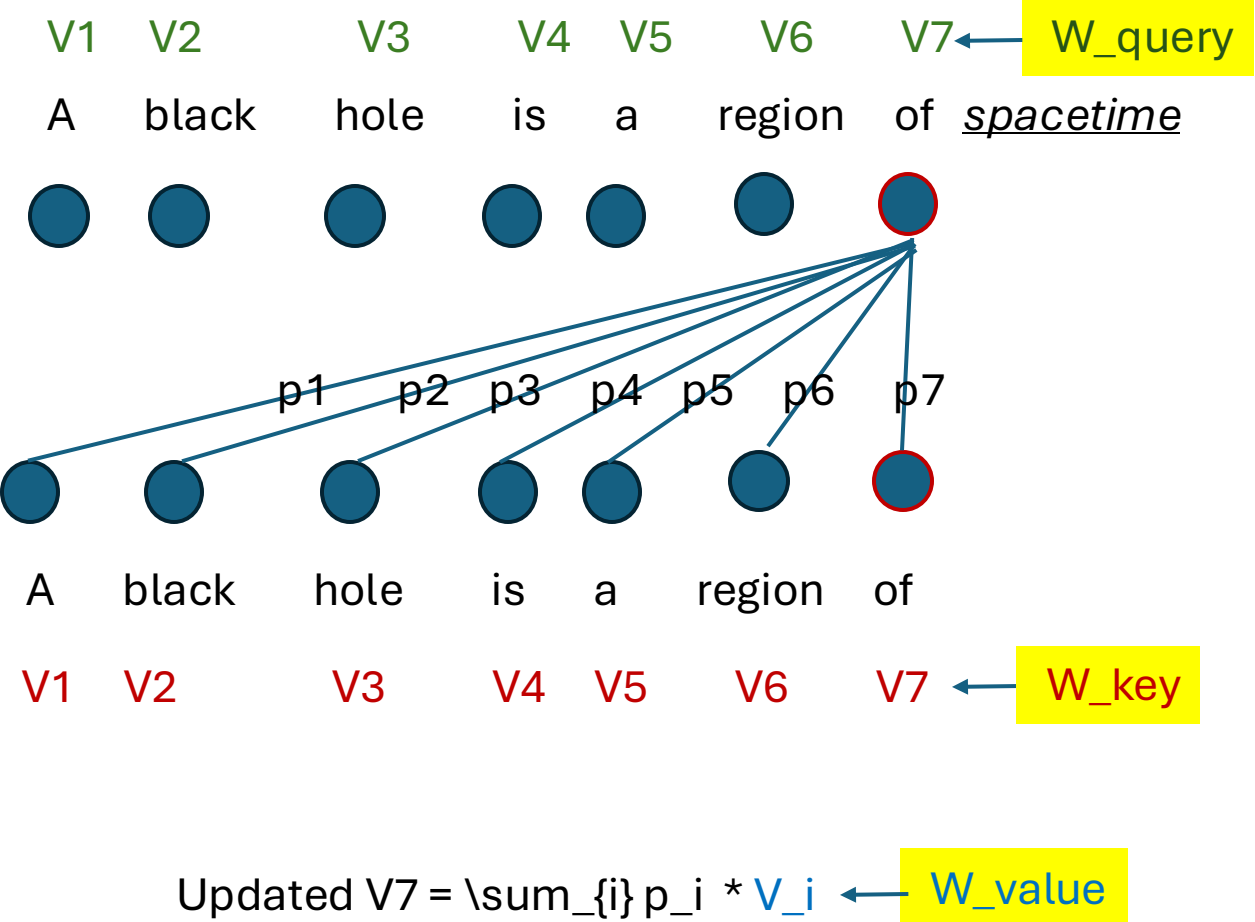
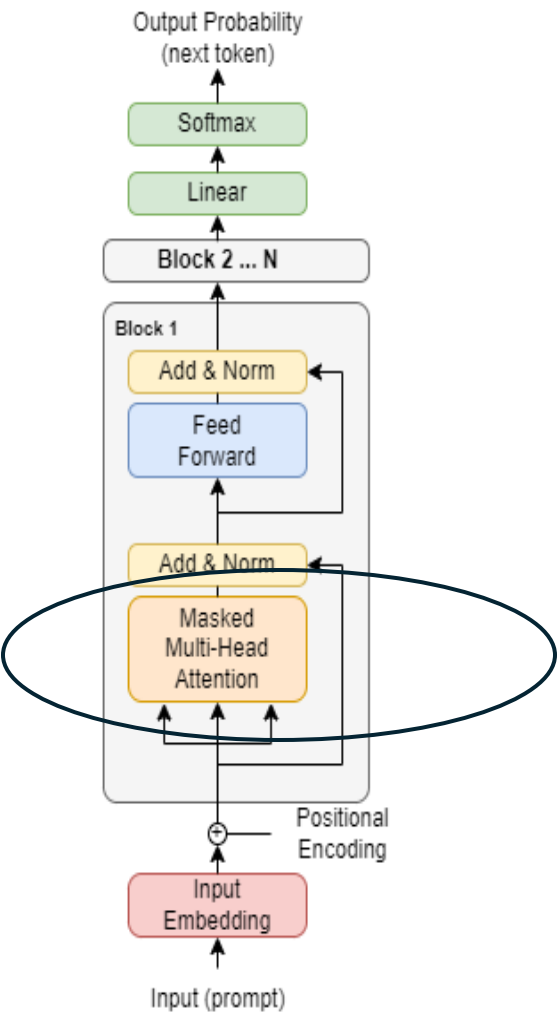
These are learnable parameters of attention function

2. Then, we compute the self-attention:

$$\text{Attention}(\text{Query}, \text{Key}, \text{Value}) = \text{softmax}(\text{Query} * \text{Key}^{\text{trans}} / \sqrt{d_k}) * \text{Value}$$



Attention Function



1. Compute dot product with all tokens before it
2. Convert these values to a probability distribution using softmax function
3. Update the representation of the word using convex combination of other tokens

Actual Self-Attention

```
B,T,C = 1,8,32 # batch, time, channels  
x = torch.randn(B,T,C)
```

```
# let's see a single Head perform self-attention
```

```
head_size = 16
```

```
key = nn.Linear(C, head_size, bias=False)
```

```
query = nn.Linear(C, head_size, bias=False)
```

```
value = nn.Linear(C, head_size, bias=False)
```

```
k = key(x) # (B, T, 16)
```

```
q = query(x) # (B, T, 16)
```

```
wei = q @ k.transpose(-2, -1) # (B, T, 16) @  
(B, 16, T) ---> (B, T, T)
```

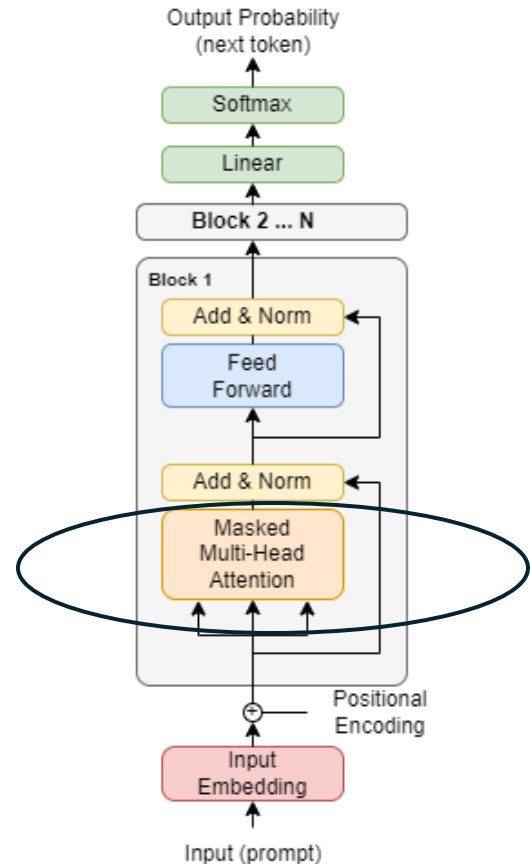
```
tril = torch.tril(torch.ones(T, T))
```

```
wei = wei.masked_fill(tril == 0, float('-inf'))
```

```
wei = F.softmax(wei, dim=-1)
```

```
v = value(x)
```

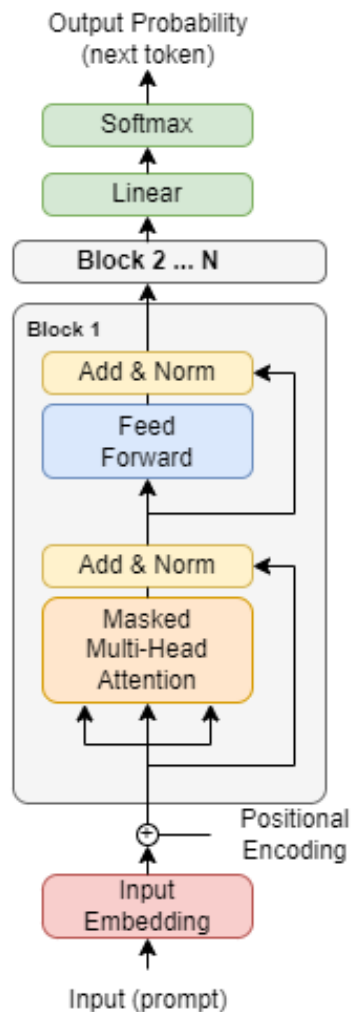
```
out = wei @ v
```



(Andrej Karpathy)

<https://www.youtube.com/watch?v=VMj-3S1tku0&list=PLAqhlrjkxbuWI23v9cThsA9GvCAUhRvKZ>

Masked Multihead Attention



3.2.2 Multi-Head Attention

Instead of performing a single attention function with d_{model} -dimensional keys, values and queries, we found it beneficial to linearly project the queries, keys and values h times with different, learned linear projections to d_k , d_k and d_v dimensions, respectively. On each of these projected versions of queries, keys and values we then perform the attention function in parallel, yielding d_v -dimensional

⁴To illustrate why the dot products get large, assume that the components of q and k are independent random variables with mean 0 and variance 1. Then their dot product, $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$, has mean 0 and variance d_k .

4

output values. These are concatenated and once again projected, resulting in the final values, as depicted in Figure 2.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

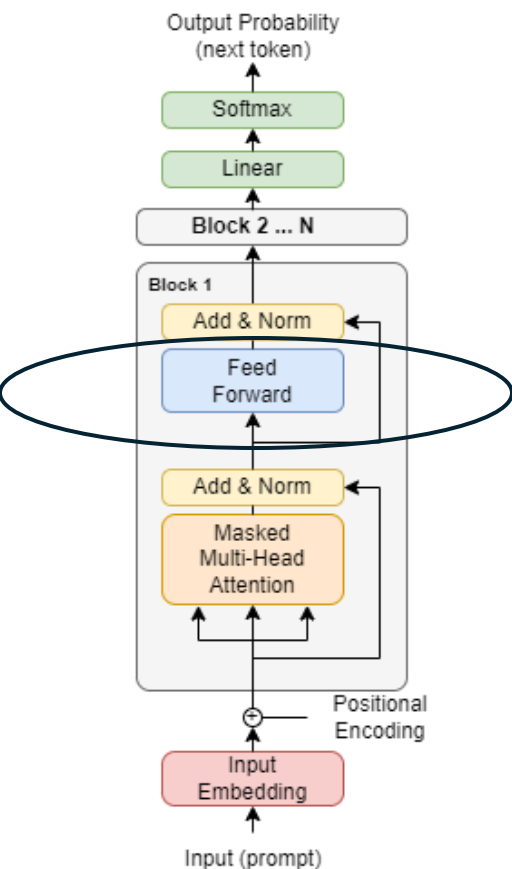
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O$$
$$\text{where } \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

In this work we employ $h = 8$ parallel attention layers, or heads. For each of these we use $d_k = d_v = d_{\text{model}}/h = 64$. Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.

Feed Forward Layer

(the same network we saw in digit recognition task)



3.3 Position-wise Feed-Forward Networks

In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (2)$$

While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convolutions with kernel size 1. The dimensionality of input and output is $d_{\text{model}} = 512$, and the inner-layer has dimensionality $d_{\text{ff}} = 2048$.

Repeat A Transformer Block Few Times

A single transformer Block:
Self-Attention Layer
Normalization Layer
Feed Forward Layer



You

how many tranformer blocks GPT4 has?



ChatGPT

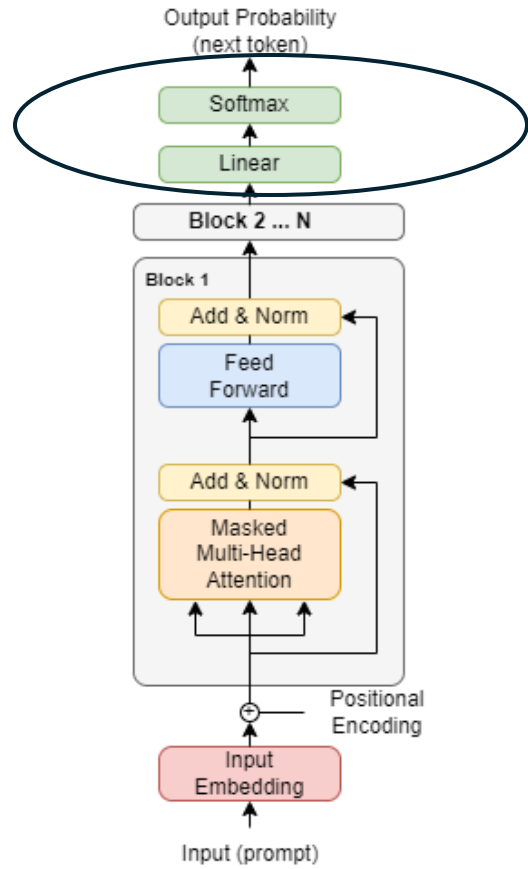
As of my last update in January 2022, the exact architecture of GPT-4 hasn't been officially disclosed, as it's a hypothetical model. However, I can provide an estimate based on the progression of the GPT series.

GPT (Generative Pre-trained Transformer) models typically consist of multiple layers of transformer blocks. GPT-3, for example, has 96 layers of transformer blocks.

Assuming that GPT-4 follows a similar architecture but with even more parameters and complexity, it could potentially have over 100 layers of transformer blocks. However, the actual number of transformer blocks in GPT-4 would depend on various factors, including research advancements, computational constraints, and the specific design choices made by the developers.

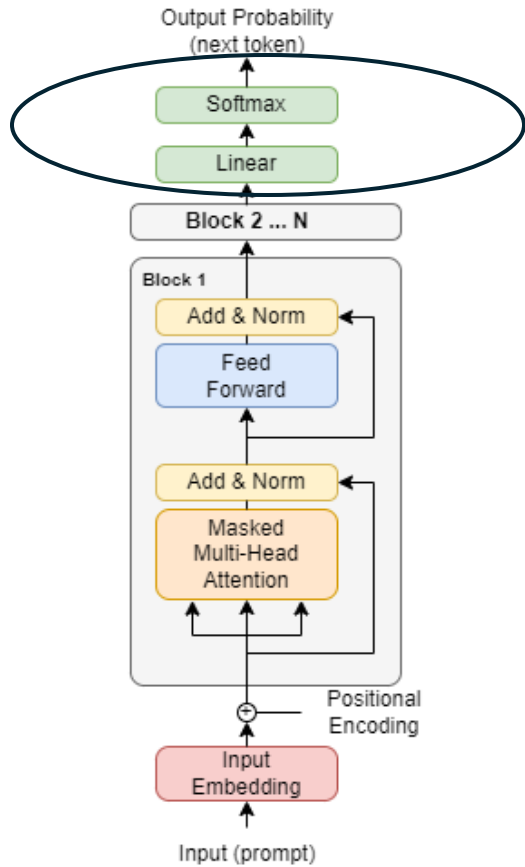
It's essential to keep in mind that without official documentation or release, any information regarding GPT-4's architecture is speculative.

Final Layer



What should this linear layer + softmax do?

Final Layer



What should this linear layer + softmax do?

It should take the vector representing a token, and output a probability distribution over the set of all possible tokens!

To do that: we learn another linear layer (matrix) and then apply softmax!

We get a distribution over the all tokens in our dictionary.

This happens for every token in the sequence.

Done.

DNNs = Matrix Multiplications => GPUs...

Every layer of a transformer (or any DNN) for that matter is

- simply matrix-matrix multiplication; it is matrix-matrix not matrix-vector because we are processing multiple examples at the same time.
- you can also take multiple sentences simultaneously, and try to predict next token in each of them; then your input will be a tensor, first index representing the sentence; tensor multiplication is similar to matrix multiplication, but libraries parallelize them on GPUs on the first dimension.
- only other operation we do is softmax and some RELU type non-linear operations.
- That's pretty much it! DNNs are surprisingly simple objects!

DNNs = Matrix Multiplications => GPUs...

Every layer of a transformer (or any DNN) for that matter is

- simply matrix-matrix multiplication; it is matrix-matrix not matrix-vector because we are processing multiple examples at the same time.
- you can also take multiple sentences simultaneously, and try to predict next token in each of them; then your input will be a tensor, first index representing the sentence; tensor multiplication is similar to matrix multiplication, but libraries parallelize them on GPUs on the first dimension.
- only other operation we do is softmax and some RELU type non-linear operations.
- That's pretty much it! DNNs are surprisingly simple objects!

Intuition of transformers

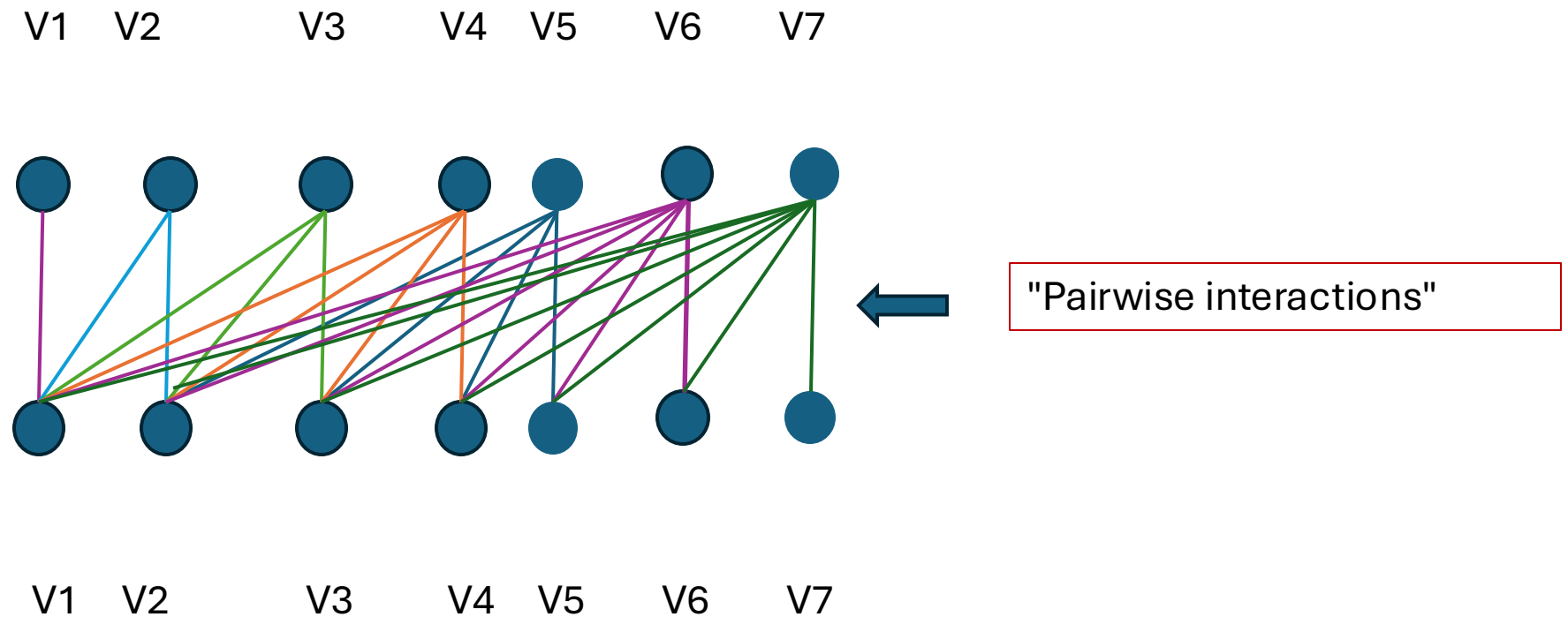
Attention Layers:

Transforming the data to get a rich representation

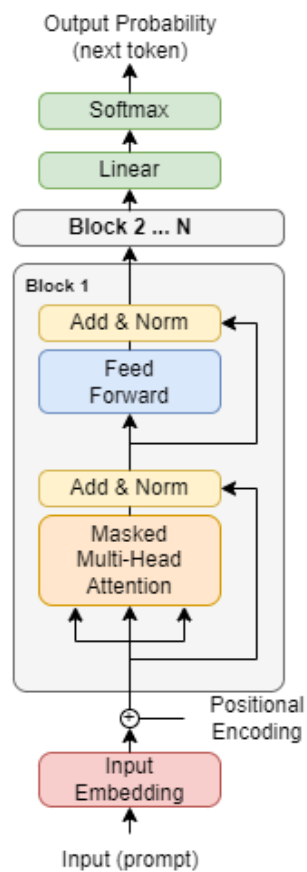
Feed Forward Layers:

Computing some function over the data

Intuition 2: *Attention Layers:*



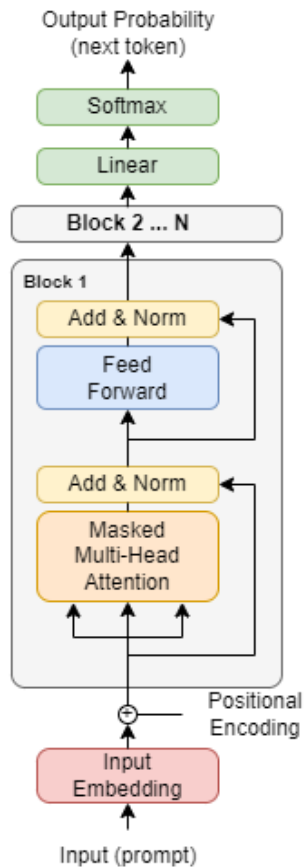
But note that attention blocks are repeated!



Which means that in every $i+1$ attention layer, each token representations get richer, and can **capture arbitrary subset of interactions!**

Repetition of attention blocks allows information "mixing"

Intuition 3: Why multiple heads?



Allow the model to capture different kinds of similarities

For same query, different tokens can become keys in different ways. So, you want to allow for such situations.

Multi-head Attention allows each token to capture *different relationships* among tokens

“Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.” - from the paper

From the paper

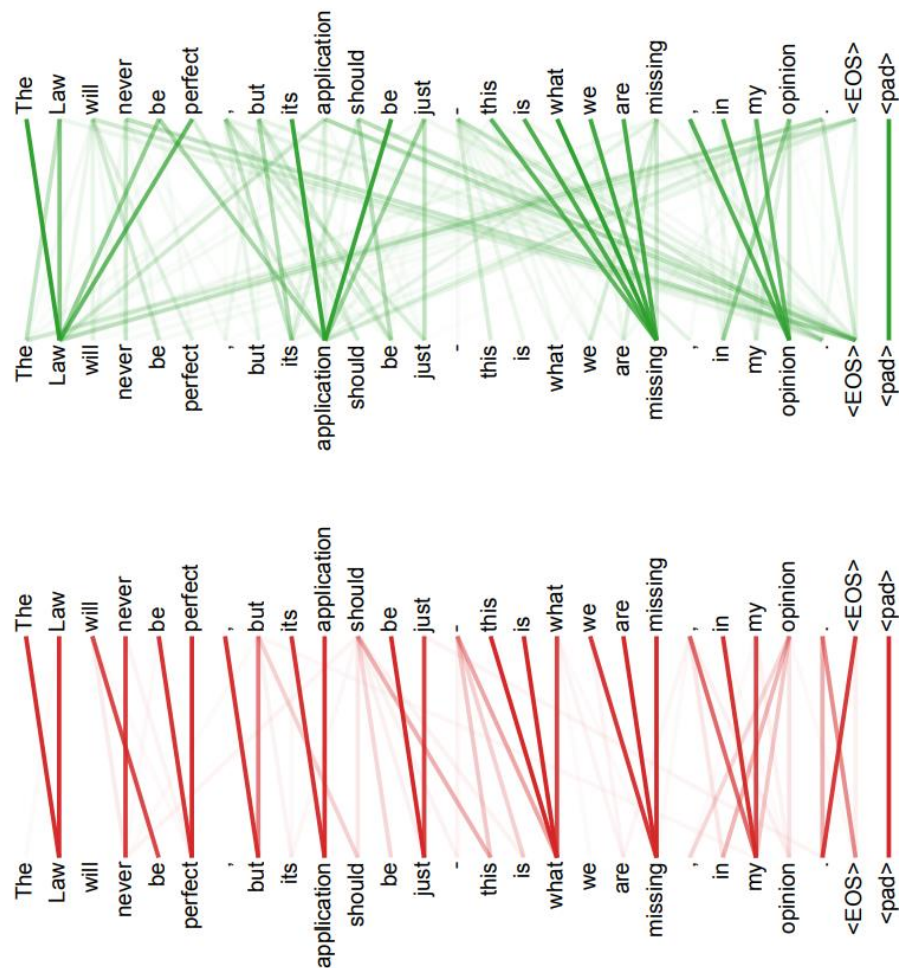


Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.

Three Breakthrough Ideas

Unsupervised or self-supervised
Learning

Transformer Architecture

Scale

See Andrej Karpathy's lecture videos